DOI: http://dx.doi.org/10.7551/978-0-262-33027-5-ch098

Towards a methodology for describing the relationship between simulation and reality

Eric Schneider[1], Elizabeth I. Sklar[1], M. Q. Azhar[2], Simon Parsons[1] and Karl Tuyls[1]

[1]Department of Computer Science, University of Liverpool, UK
{eric.schneider, e.i.sklar, s.d.parsons, k.tuyls}@liverpool.ac.uk
[2]Borough of Manhattan Community College and Graduate Center,
City University of New York, New York, USA
mazhar@bmcc.cuny.edu

Abstract

For research that carries out experiments in simulation, an important question is how the results will translate into the real world. This paper proposes a method for comparing results obtained in simulated versus physical environments, based on *interval* relationships between metrics gathered in both settings. The approach is motivated by the fact that the relationship between absolute measures often does not tell much. For example, the amount of time taken to complete a task in simulation versus the same task in the physical world could always be shorter in simulation because of speed-up factors embedded in the simulator. Three different metrics are introduced that describe different interval relations, and these are demonstrated using two case studies.

Introduction

It is common practice in *artificial life*, *evolutionary computation*, *multiagent systems* and *robotics* to employ *simulation* as a means to evaluate an approach which is intended to be deployed in some type of *real* environment. "Real" might be physical (as in the case of robotics) or might be interactive (as in the case of human-agent systems) or might be real-time (as in the case of systems that respond to live data feeds, such as financial stock prices, or sensors, such as traffic lights). The advantage of simulation over reality is that we typically have more control over and easier access to the simulated environment. This implies that it is simpler to test algorithms, or whatever we are working on, in the simulated environment first—i.e., before it is deployed in reality.

Notwithstanding the many issues in transferring results from simulation to reality (Brooks (1992)), the general wisdom is that "if it works in simulation", then it will work, to some degree, in reality; and if it doesn't work in simulation, then it certainly will not work in reality. While there is a reasonable literature on the notion of *verification*, especially in multiagent systems, and some attention paid to the notion of *validation* in multiagent-based simulation (though not enough, in our opinion), these pieces of work do not attempt to measure **how good** a simulation is with respect to the reality it is meant to approximate and and **how well** the simulation solution will **transfer** to reality. In other words, if we say that testing in simulation will ensure that a particular approach will work in reality *to some degree*, what does that mean? To *what* degree? And what is a *degree*?

In the work presented here, we address exactly those questions. Our contention is that a simulation environment will never fully or exactly emulate everything that happens in a real environment (agreeing with Brooks (1992)), but if we have some structured way of measuring and describing what the degree of closeness is, then we will have a structured way of being able to define how robust our approach—tested in simulation—is with respect to reality. Especially in cases where testing in reality is risky (e.g., nuclear cleanup) and/or expensive (e.g., planetary exploration), it would be very useful to know how much we are gaining by the knowledge obtained in the simulation environment. This work is particularly relevant in the *artificial life* community because our methodology can be applied to assess the utility of *artificial* approximations or imitations of real phenomena.

This paper is organised as follows. The next section highlights prior work on describing the relationship between real and simulated environments. Then we describe our methodology in abstract terms, and outline an example in which we applied our methodology to some of our own work in robotics and multiagent systems. Finally, we close with some discussion and conclusions.

Related Work

As it is often infeasible to develop robot behaviours on physical hardware, there has been a good deal of investigation into developing behaviours in simulations. An early example of this is Koza (1991), which used genetic programming to recreate the kind of navigation that Mataríc (1990) had hand-coded, and led some to conclude that it would be straightforward to use evolutionary techniques to learn robot controllers that could be dropped into real robots that would then operate as desired in the real world.

Responding to this position, Brooks (1992) raised concerns about the transferability of behaviours learned in simulations due to significant differences between simulation and physical environments.

Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, Karl Tuyls (2015) Towards a methodology for describing the relationship between simulation and reality. Proceedings of the European Conference on Artificial Life 2015, pp. 562-569

First, working purely in simulation, that is without regularly checking the results of the simulation against what happens in the real world, could lead to evolutionary techniques focusing on problems that just don't exist in the real world. Second, if simulators do not accurately model the errors that occur in sensing and actuation, evolutionary techniques that evaluate their output only in simulation are unlikely to evolve controllers that will work on real robots.

Jakobi et al. (1995) introduced the term *reality gap* to describe the differences between reality and simulation that Brooks had described, and went on to provide evidence both of the existence of the gap and of the possibility of overcoming it. They evolved controllers under three conditions: no noise, noise equivalent to that measured in the real world ("observed noise" in their terminology), and much more noise that is observed in reality. Controllers evolved with observed noise worked when transferred onto a real robot. Controllers evolved either with no noise, or with much more than observed noise, failed on real robots.

There have been efforts to skirt the reality gap rather than model it explicitly. Vaughan and Zuluaga (2006) propose using a simulator at various points during the performance of a physical task—selecting targets and planning paths to them—in order to find viable solutions, and especially to avoid dangerous outcomes like colliding with walls or other obstacles. This use of simulation is similar to real-time planning methods like the Dynamic Window Approach (Fox et al., 1997) to collision avoidance. More recently, Farchy et al. (2013) used "Grounded Simulation Learning" to optimize a walk cycle on a humanoid robot. "Grounding" involved learning a controller via a small number of trials on a physical robot before refining the controller through a much larger number of trials in a simulator. Further development occurred over a number of round trips through this process. Rather than tune the simulator to match observed noise in the robot's physical environment, the *behaviour* of the simulated robot was constrained to match real world results.

Marques and Holland (2009) define architectures for "functional imagination" for simulation, that is, architectures in which behaviours developed in simulation are transferable to physical implementations in some useful way. They identify a set of necessary and sufficient features for a simulator to provide "behavioural benefit" to physical performance, but do not directly address the problems raised by the reality gap, much less how to measure or overcome it.

Koos et al. (2010) note that transferability and efficient performance in a simulator, which may exploit bugs or poor models of a physical environment, are conflicting goals. They propose an evolutionary algorithm that aims to optimize for both objectives. To help achieve this, they define a "simulation-to-reality disparity factor" for controllers developed in simulation. This factor is based on the differences in the controller's performance observed in simulation and physical environments according to certain "behavioural features". Examples of features are distance covered during an experiment or the angular orientation of a robot and the end of its behaviour.

Approach

The question of how well simulation predicts performance in a physical environment can be examined in a number of ways. A simple method is to select a particular metric, e.g., *distance travelled* in a mobile robot domain, and compute that measurement in the robot's physical environment as well as in a simulated environment with the same geometric specifications. In earlier work (Sklar et al., 2012), we did just this. We selected six metrics and ran point-to-point comparisons between the physical environment and a parallel simulated environment. Our results showed that, while the individual metrics—scalar values, i.e., single *points* within a distribution—do not line up in absolute terms, they do align in relative terms. We had the idea that the *relationships* between metrics could be expressed as some *function* that could be computed from sample results collected in physical and simulated environments. In this way, one could collect a statistically significant sample in simulation, and then apply the function to those results and obtain a fair approximation of what the physical results would be. For example, we could train a neural network to predict the physical results based on simulated results, using our sample data set as the training set.

Here, we continue this line of inquiry, but propose three additional ways of looking at the relationships between simulated and physical results:

- First, instead of performing point-to-point comparisons, we examine the distribution of values for a particular metric and compute an *interval* that describes the bounds of the distribution (such as $\mu \pm \sigma$ or $max - min$ value range). We can then compare the intervals for individual metrics, in simulation versus physical environments.

- Second, in addition to considering intervals for individual metrics, we compare the intervals for *sets* of metrics.

- Third, we consider *rank order* comparisons of groups of metrics. Since simulation is often used to assess the impact of various experimental conditions, we hypothesize that the best-to-worst ordering of each condition can be compared in physical and simulated environments and the ordering itself can be useful even if the point-to-point or interval relationships do not align.

We believe that these comparisons are useful additions to the point-to-point comparisons in describing the predictive power of a simulation environment with respect to a parallel physical environment. For example, rather than saying that simulation results predict physical results "to some degree", we can say that the results are comparable with respect to

Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, Karl Tuyls (2015) Towards a methodology for describing the relationship between simulation and reality. Proceedings of the European Conference on Artificial Life 2015, pp. 562-569

relations		*picture*
$x < y$	$y < x$	XXX YYY
$x = y$		XXX YYY
x meets y	y meets x	XXXYYY
x overlaps y	y overlaps x	XXX YYY
x during y	y during x	XXX YYYYYY
x starts y	y starts x	XXX YYYYY
x finishes y	y finishes x	XXX YYYYY

Figure 1: Allen's 13 temporal interval relationships between two time periods, x and y (from Allen (1983)).

statistical intervals or *rank ordering*. Next, we describe each methodology in detail.

Statistical interval comparison

Allen (1981, 1983) describes temporal relations between events and identifies thirteen possible relationships between any pair of time intervals. These are listed in Figure 1.

We apply the same idea to any scalar metric that can be expressed as a *statistical interval*. For example, it could be a *confidence interval*, centered on the mean and bounded by $\pm n$ standard deviations; or it could be a *quartile* interval; or a *min-max* interval. Applying the statistical interval relationship method works as follows:

1. Collect a set of raw data for a particular metric in the simulated environment—a statistically significant sample—and another set of data for the same metric in the corresponding physical environment (a smaller sample).

2. Compute the mean, μ, and standard deviation, σ, of both samples (note that we assume that the distribution in the physical environment will be normal, thus the mean and standard deviation are still valid, albeit not as reliable as when the data set is larger).

3. Plot the interval $[\mu - n\sigma, \mu + n\sigma]$ for both simulated and physical data sets, as two vertical columns in a 2-dimensional graph. We select n based on the percentage of the distribution that we want our interval to cover. For example, $n = 1$ will cover 68% of the distribution and $n = 2$ will cover 95% of the distribution[1].

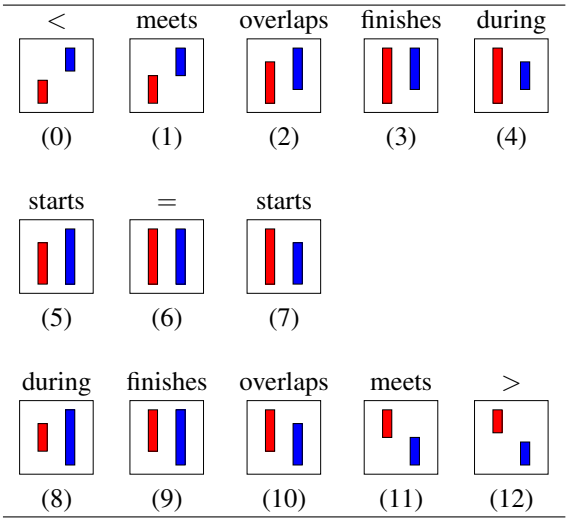


Figure 2: Statistical intervals, using Allen's relations labels. See text for explanation.

Now we can examine the relationships between these intervals, as illustrated in Figure 2. The relations are labelled using Allen's terminology, but the order in which the relations are displayed is sorted so that the equality relationship is in the middle and the further we go from the middle, the more disparity between the values being compared. Cases (0) and (12) are when the data is completely unaligned and one set of values is strictly less (or greater) than the other. Cases (1) and (11) are when one set of values is less (greater) than or equal to the other. Cases (2) and (10) are when the two sets of values overlap. Cases (3) and (9) are when the upper bounds of both sets are equal, but not the lower bounds. Cases (4) and (8) are when one set of values is completely contained in the other set. Cases (5) and (7) are when the lower bounds of both sets are equal, but not the upper bounds. Case (6) is when the data is perfectly aligned.

We use the statistical intervals to compare experimental results under two different conditions. These could be *physical* versus *simulated*, which is what we are particularly interested in here; however, these could generalise to comparing other pairs of experimental conditions. For example, consider Case (8). This case indicates that the performance in one metric (blue) "contains" the performance of the other (red). This implies that performance under the blue condition is more variable than under the red condition. If we are comparing performance in, say, simulation in the left-hand (red) interval and physical in the right (blue), we might be able to say that performance (in this metric) was more variable in the physical setting. If our goal is to derive behaviours in simulation (red) that are guaranteed to fall within

[1] These are standard values for normal distributions.

a certain interval in the physical world (blue), then we cannot make this claim if the data matches Case (8); however, if our data (red:simulation; blue:physical) matches Case (4), then we can make the claim.

Alternatively, we might be interested in comparing metrics resulting from experimental conditions that differ in ways other than simulated versus physical. For example, we might be interested in comparing how a robot interacts with a person when the robot is programmed using two different behaviours, called `beh1` and `beh2`. Experimental results could be obtained from people interacting with a physical robot that exhibits both behaviours, as well as with a simulated robot that also exhibits both behaviours. If the statistical interval relationship for one metric resulting from `beh1` (red) compared with `beh2` (blue) falls into Case (2) for the physical robot, and does the same for the simulated robot, then we can be confident that the simulation environment produces results reliably similar to the physical environment in order to be able to use the simulated environment for evaluating this particular metric.

Statistical interval set comparison

Our statistical interval comparison provides a structured way of describing the relationship between the values obtained under two different experimental conditions of an *individual metric*. Typically, though, experimental results examine more than one metric. Thus, we define a *statistical interval set* methodology with which to compare the performance under two different experimental conditions of a *set of metrics*. In particular, it is useful to know how *consistent* the relationships are from one metric to another in a set. For example, we might be able to say that all metrics in the set which measure "time" are Case (0), but all metrics in the set which measure "distance" are Case (12).

Rank ordering comparison

Another way we look at the relationships between physical and simulation is to examine the *rank ordering* in values of an individual metric obtained under a set of different experimental conditions. Take again the example of distance travelled. Supposed we want a robot to visit ten points in its environment, and we have five different ways of deciding the order in which the robot visits the points. Let's call these `v1` through `v5`. We run experiments in both simulated and physical environments, and we compute the distance travelled for all five visiting methods. Then we sort the distance values, from shortest to longest, and obtain a rank-order for the corresponding visiting methods. We can do this for experiments conducted both in simulation and in the physical environment. If the rank-ordering is the same between the simulated and physical environments, then we can be confident that simulation is an effective method for comparing experimental conditions along the metric chosen. For example, in the sample human-robot experiment described above, if the distance travelled for the robots using `v1` is the shortest and the distance travelled for the robots using `v3` is the longest, in both physical and simulated environments, then we can be confident that the simulation environment produces results reliably similar ot the physical environment in order to be able to use the simulated environment for evaluating this particular metric across this set of behaviours.

Case Studies

We demonstrate the utility of our methodology with two case studies. The first case study involves a series of experiments that evaluate several different task allocation mechanisms for a multi-robot team. The second case study involves a series of experiments that evaluate two different mechanisms for interaction in a human-robot scenario. First we describe each case study, and then apply our four comparison methods to each: point-to-point comparison, statistical interval comparison, statistical interval set comparison and rank ordering comparison.

Case Study 1: Multi-robot task allocation

This case study involves a team of robots tasked to visit a number of *target points* in a constrained arena, organised such that one robot visits each point once. Our research in this case study concerns assessment of a number of different mechanisms by which tasks are allocated to robots. The results, with respect to allocation mechanisms, have been presented elsewhere (Özgelen et al., 2013; Schneider et al., 2014). Here, we are concerned with the comparison of results obtained in parallel *physical* and *simulated* settings.

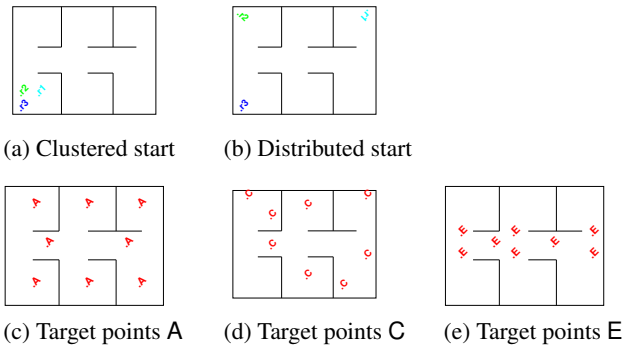
Our experimental testbed employs a dual system architecture, based on *Player/Stage*[2] (Gerkey et al., 2003; Vaughan and Gerkey, 2007), in which both physical and simulated environments share common underlying system components. The details of our framework have been described elsewhere (Sklar et al., 2011).

There are two primary differences between the physical (Player) and simulated (Stage) instantiations of our framework: one is with respect to *localisation* and the other is with respect to robot *driving*.

Localisation refers to robots knowing where they are in their environment, in terms of a coordinate-based frame of reference. In the physical setup, this information is provided by a network of cameras, suspended above the arena, which track the robots and report their (x, y, θ) positions to all team members, through a central server process. In contrast, in the simulation setup, localisation is "perfect" because the simulator knows where all the robots are at all times. Thus, the physical environment is more *noisy* with respect to robots knowing where they are. *Driving* refers to robots knowing how to move, i.e., which motor(s) to turn on

[2] `http://playerstage.sourceforge.net/`

Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, Karl Tuyls (2015) Towards a methodology for describing the relationship between simulation and reality. Proceedings of the European Conference on Artificial Life 2015, pp. 562-569



(a) Clustered start (b) Distributed start

(c) Target points A (d) Target points C (e) Target points E

Figure 3: Scenario definitions. (a) and (b) show starting locations. (c), (d), and (e) show target point set locations.

for how long. In the physical setup, a robot *controller* process communicates abstract motion commands (e.g., "forward") to a second *driver* process which converts the abstract motion command to platform-specific byte codes and transmits the codes to the physical robot. This abstraction of motion commands means that the only platform-specific element of the system is down at the driver level. In contrast, in the simulation setup, the same robot controller process sends the abstract motion commands to a robot driver in the simulator. Thus, the physical environment is again noisier than simulation and is also slower, because there is an extra level of communication (from the driver to the physical robots) that does not exist in simulation.

The experiments we conducted measured results in six different scenarios. All scenarios involved $n = 3$ robots and $m = 8$ target points. There were two sets of starting locations, one "distributed" and one "clustered", and three sets of target point locations (A, C and E). The starting locations and the target points can be seen in Figure 3. Experiments were conducted with each scenario using each of four different task allocation mechanisms. Here, we will refer to these generically as TAM1 through TAM4, because detailing the mechanisms is not the point of this paper (as above, details were reported elsewhere). Each combination was run 6 times in the physical environment and 30 times in the simulation environment.

$$
\begin{aligned}
\textit{start location} &: \{\text{Clustered, Distributed}\}\times \\
\textit{targetpoint set} &: \{\mathsf{A}, \mathsf{C}, \mathsf{E}\}\times \\
\textit{mechanism} &: \{\mathsf{TAM1}, \mathsf{TAM2}, \mathsf{TAM3}, \mathsf{TAM4}\}\times \\
\textit{environment} &: \begin{cases} \text{6 physical} & = 144 \text{ runs} \\ \text{30 simulation} & = 720 \text{ runs} \end{cases}
\end{aligned}
$$

For each run of each experiment, we recorded the following 6 metrics: (1) *deliberation time*, the total time required to allocate the target points; (2) *execution time*, the total time required to visit all the target points; (3) *distance travelled*, the total distance travelled by all robots to visit their assigned target points; (4) *idle time*, the total amount of time that robots were not executing a task, i.e., because they had no (more) target points to visit; (4) *delay time*, the amount of time robots spent avoiding collisions with others (explained below); and (6) *near collisions*, the number of times robots detected another's presence and stopped to negotiate right-of-way. Because each experiment involves many robots moving in a restricted area, they naturally get in each other's way. When robots are close enough to require evasive action, our system detects a "near collision," and the robots stop moving. Then the robot closest to its goal (current target point) is given the right-of-way. The other robot waits until its path is clear, and then continues on its way. The time that a robot was stopped for this reason is its *delay time*. In addition to measuring delay time, we counted the number of times that robots were delayed in this manner (*near collisions*).

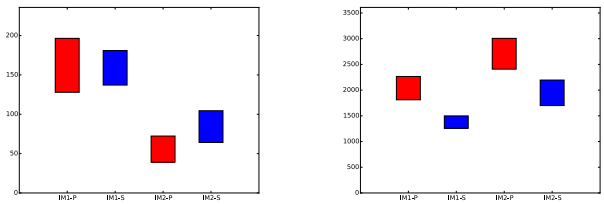
Case Study 2: Human-robot interaction

This case study involves a human-robot team collaboratively looking for "treasures" hidden in an environment that the robot can explore but the human cannot enter. In order for the team to complete the task—finding and correctly identifying all the treasures—the human and robot have to work together. There are tasks that only the robot can perform, such as wandering around in the environment and capturing images of what it "sees" there; and there are tasks that only the human can perform, such as identifying a particular treasure within an image (which is provided by the robot). Our research in this case study concerns assessment of an *argumentation-based dialogue* mechanism for facilitating human-robot collaboration. The results, with respect to interaction mechanism, have been presented elsewhere (Azhar et al., 2013; Sklar et al., 2013). Here, we are concerned with the comparison of results obtained in parallel physical and simulated settings.

The experimental setup is similar to that employed by Case Study 1. The robot explores the same physical environment as the multi-robot team. The physical robot is implemented using Player and the simulated version of the robot is implemented in Stage. The experiments we conducted measured not only the same 6 performance metrics described for Case Study 1, but also a number of metrics that assess the usability and impact of two interaction mechanisms. Here, we will refer to these generically as IM1 and IM2, because detailing the mechanisms is not the point of this paper (as above, details were reported elsewhere). We conducted a user study for Case Study 2, which involved 60 human subjects: 27 collaborated with a physical robot, and 33 collaborated with a simulated robot.

Results

Having presented each of our cases studies, here we apply our method for comparing the results of experiments to both



(a) Deliberation time (b) Total distance

Figure 4: Deliberation time and distance travelled for the human-robot interaction case study. The left plot shows deliberation time, the right plot shows distance travelled. Each plots compares results for physical (red) and simulated (blue) across both forms of dialogue.

case studies in turn. We start with the human-robot interaction scenario since it is simpler.

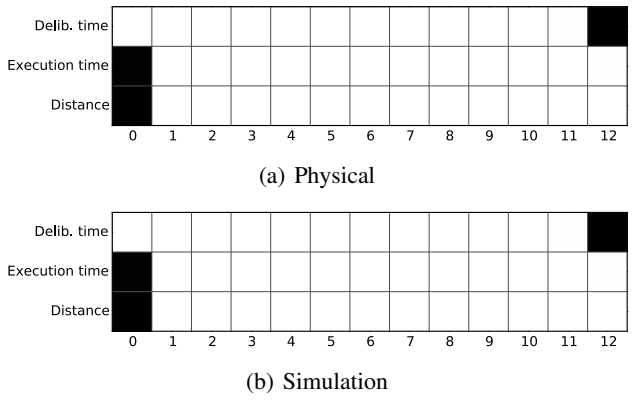
Human-robot interaction

Figure 4 shows the statistical interval method applied to the human-robot interaction scenario for two metrics, deliberation time and the total distance travelled. In this figure, we compare results obtained using physical robots against results obtained in simulation (red vs blue bars) to see if there are systematic differences. We see that for *total distance travelled*, the results obtained fall into interval relationship Case (12), where the results in simulation are consistently less than those on physical robots (and since we are using twice the standard deviation to construct the intervals, we can conclude that this difference is significant). For *deliberation time*, the results for IM1 fall into relationship Case (4) with the simulation results contained within the physical results (so any result in simulation is within what is found in practice) whereas the results for IM2 fall into relationship Case (2).

Figure 5 illustrates our "sets of intervals" analysis. Here for each metric, we compare experimental conditions—i.e., the two interaction mechanisms.

The key difference between this method and the method demonstrated above is that the statistical interval method (above) compares the red vs blue bars for each experimental condition, whereas here, the statistical interval set method compares the relationship between the two red bars vs the relationship between the two blue bars. Each cell in the heatmap in Figure 5 contains the number of pairs of results (i.e., two red bars for the physical plot at the top) which fall into each interval relationship case (numbered 0..12 across the x-axis of the plot). In this case, there is only one comparison (between the two conditions), so there is only one entry in each row in the heatmap.

But the idea of the heatmap is to make it easy to spot correspondences between physical and simulation results. Figure 5 shows that simulation and physical agree exactly on



(a) Physical

(b) Simulation

Figure 5: Heatmap showing sets of interval relationships for the task allocation case study. Darker values represent higher counts.

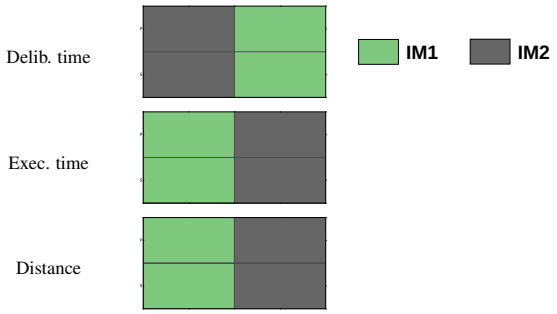


Figure 6: Rank order of metrics for the human-robot interaction case study. Values are ordered from left to right.

the comparative results for the two interaction mechanisms.

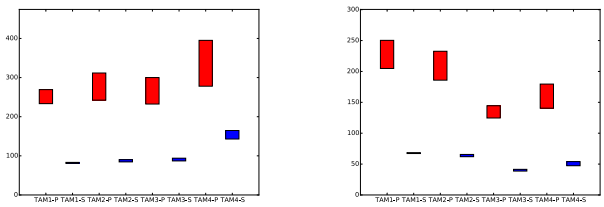
Figure 6 provides a way of looking at the rank order over metrics. These are rank-ordered, so that the top-valued condition is in the left column and the bottom-valued condition is in the right column. Since there are only two experimental conditions, this is not a complex plot; but as above, the idea is to make it easier to spot correspondences between physical (top row in each plot) and simulation (bottom row). Figure 6 shows perfect alignment in rank-ordering for the metrics illustrated.

Multi-robot task allocation

Figure 7 shows the comparison of the individual statistical intervals for one of the metrics and all the task allocations mechanisms. Here we use two standard deviations to define the intervals. In this particular case, it is easy to see that for this metric there is a consistent relationship between physical and simulated results across both start conditions. In all cases, the relationship between the intervals is Case (12), which allows us to say that the execution time for simulations is significantly less than that for physical robots.

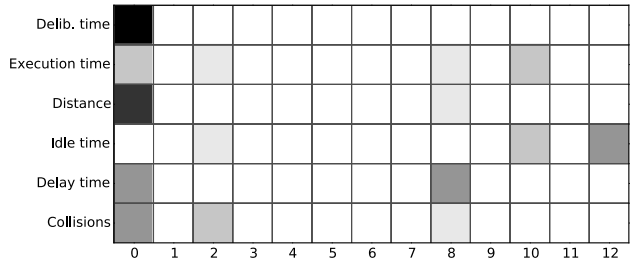
Figure 8 shows the sets of intervals in heatmap form, a

Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, Karl Tuyls (2015) Towards a methodology for describing the relationship between simulation and reality. Proceedings of the European Conference on Artificial Life 2015, pp. 562-569

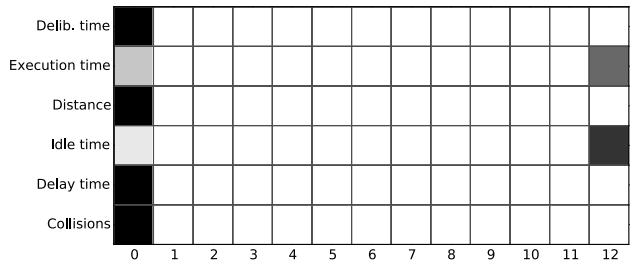


(a) Clustered start (b) Distributed start

Figure 7: Execution time for the task allocation case study. The left plot shows the clustered start condition, the right plot shows the distributed start condition. Each plots compares results for physical (red) and simulated (blue) across all four task allocation mechanisms.



(a) Physical



(b) Simulation

Figure 8: Heatmap showing sets of interval relationships for the task allocation case study. Darker values represent higher counts.

more complex picture than in Figure 5. Figure 8 (a) summarises all the results on physical robots for two task allocation mechanisms, and (b) summarises all the results in simulation for the same two mechanisms. Each row in the heatmaps summarises all the results for a single metric across the different start configurations and the different sets of task points. For each experiment we generate the statistical intervals, establish which of the interval relationships that they fall into, and then count how many experiments stand in each of the thirteen interval relationships. Each cell then displays the relevant count, with darker cells reflecting a higher count.

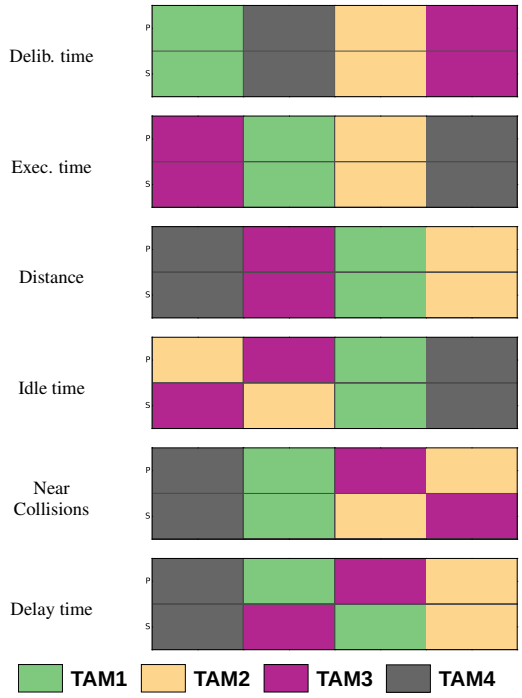


Figure 9: Rank order of metrics for the task allocation case study. Values are ordered from left to right.

What we are looking for here is similarity between physical and simulated experiments. For these two mechanisms, we can see strong agreement in terms of deliberation time: both physical and simulation have all comparisons falling into Case (0); good agreement on distance: all of simulation and most of physical fall into Case (0); and lesser agreement on delay time and collisions: simulated results all fall into Case (0), physical are split between Cases (0), (4) and (6); and idle time: all simulated are Case (3), while physical are split between Cases (3) and (7). These relationships allow us to identify when simulations will be good predictors of behaviour on real robots.

These relationships between mechanisms are shown even more clearly in Figure 9, which shows the application of the idea of rank-ordering to the results. In particular, this figure shows, for each metric, the relative performance of each mechanism across environments. Rankings are indicated by placement in the graphs, with the top-valued experimental condition (task allocation mechanism, in this case) on the left and the worst performer on the right. There are two rows for each metric: the top row indicates ranks for runs performed in the physical environment while the bottom indicates those for thesimulation environment. These results show quite clear agreement of rank orders in physical and simulation environments. Deliberation time, execution time, and distance travelled show exact agreement. The remaining metrics–idle time, delay time, and number of near collisions–are "misaligned" by at most one rank order.

Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, Karl Tuyls (2015) Towards a methodology for describing the relationship between simulation and reality. Proceedings of the European Conference on Artificial Life 2015, pp. 562-569

Summary and Future Work

The interval relationships developed in this work, and the corresponding graphical representations give an immediate and, at the same time, nuanced indication of the way physical and simulation environments relate to one another. We have presented three different ways of comparing results obtained in parallel physical and simulation environments, and we have demonstrated these methods of comparison in two different case studies.

There are a number of open issues that have emerged as result of our preliminary work presented here. First, it is a logical question to ask what the methods presented here provide that traditional *t*-tests from statistics do not tell us. Second, another natural question to ask is how the methods presented here hold up in the face of significant stochasticity in results (e.g., where 30 runs does not offer convergence or demonstrate normal distributions), as well as handling of outliers. Finally, the methodology presented examines scalar values that vary from run to run, but do not trend over time. In the case of domains that involve learning, such as any attempt to model human behaviour, certain statistics will improve as the human learns. This kind of situation, where the metric being compared is a function rather than a scalar value, will require different treatment. These questions will be investigated in future work.

Acknowledgements

Thanks to Gal Kaminka who first suggested Allen's intervals in connection with our work. Part of this work was completed while some of the authors were with Brooklyn College and Hunter College of the City University of New York (CUNY). We gratefully acknowledge the work of our CUNY colleagues including Ofear Balas and Michael Squitieri who contributed to the software development and running of some of the experiments contained this paper. This work was partially supported by grant #IIS-11-16843 from the US National Science Foundation (NSF).

References

Allen, J. F. (1981). An interval-based representation of temporal knowledge. In *7th International Joint Conference on Artificial Intelligence*, pages 221–226.

Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26(11).

Azhar, M. Q., Schneider, E., Salvit, J., Wall, H., and Sklar, E. I. (2013). Evaluation of an argumentation-based dialogue system for human-robot collaboration. In *Workshop on Autonomous Robots and Multirobot Systems*, St Paul, MN, USA.

Brooks, R. A. (1992). Artificial life and real robots. In *1st European Conference on Artificial Life*, pages 3–10. MIT Press.

Farchy, A., Barrett, S., MacAlpine, P., and Stone, P. (2013). Humanoid robots learning to walk faster: From the real world to simulation and back. In *12th International Conference on Autonomous Agents and Multiagent Systems*, pages 39–46.

Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33.

Gerkey, B., Vaughan, R. T., and Howard, A. (2003). The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *11th International Conference on Advanced Robotics*.

Jakobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in Artificial Life*, pages 704–720. Springer.

Koos, S., Mouret, J.-B., and Doncieux, S. (2010). Crossing the reality gap in evolutionary robotics by promoting transferable controllers. In *12th Annual Conference on Genetic and Evolutionary Computation*, pages 119–126. ACM.

Koza, J. R. (1991). Evolving emergent wall following robotic behavior using the genetic programming paradigm. In *First European Conference on Artificial Life*, pages 110–119, Cambridge, MA. MIT Press.

Marques, H. G. and Holland, O. (2009). Architectures for functional imagination. *Neurocomputing*, 72(4-6):743–759.

Mataríc, M. J. (1990). A distributed model for mobile robot environment-learning and navigation. Technical report, MIT Artificial Intelligence Laboratory.

Özgelen, A. T., Schneider, E., Sklar, E. I., Costantino, M., Epstein, S. L., and Parsons, S. (2013). A first step toward testing multiagent coordination mechanisms on multi-robot teams. In *Proceedings of the Workshop on Autonomous Robots and Multirobot Systems*.

Schneider, E., Balas, O., Özgelen, A. T., Sklar, E. I., and Parsons, S. (2014). An Empirical Evaluation of Auction-based Task Allocation in Multi-Robot Teams (Extended Abstract). In *13th International Conference on Autonomous Agents and Multiagent Systems*, Paris, France.

Sklar, E., Ozgelen, A. T., Munoz, J. P., Gonzalez, J., Manashirov, M., Epstein, S. L., and Parsons, S. (2011). Designing the HRTeam framework: Lessons learned from a rough-and-ready human/multi-robot team. In *Workshop on Autonomous Robots and Multirobot Systems*, Taipei, Taiwan.

Sklar, E., Ozgelen, A. T., Schneider, E., Costantino, M., Munoz, J. P., Epstein, S. L., and Parsons, S. (2012). On transfer from multiagent to multi-robot systems. In *Workshop on Autonomous Robots and Multirobot Systems*, Valencia, Spain.

Sklar, E. I., Azhar, M. Q., Parsons, S., and Flyr, T. (2013). A Case for Argumentation to Enable Human-Robot Collaboration (Extended Abstract). In *12th International Conference on Autonomous Agents and Multiagent Systems*, St Paul, MN, USA.

Vaughan, R. and Zuluaga, M. (2006). Use your illusion: Sensorimotor self-simulation allows complex agents to plan with incomplete self-knowledge. In *From Animals to Animats 9*, volume 4095 of *Lecture Notes in Computer Science*, pages 298–309.

Vaughan, R. T. and Gerkey, B. (2007). Really Reusable Robot Code and the Player/Stage Project. In Brugali, D., editor, *Software Engineering for Experimental Robotics*. Springer.

Eric Schneider, Elizabeth I. Sklar, M. Q. Azhar, Simon Parsons, Karl Tuyls (2015) Towards a methodology for describing the relationship between simulation and reality. Proceedings of the European Conference on Artificial Life 2015, pp. 562-569