Behavioural and Structural Imitation Models in Facebook's WW Simulation System

J. Ahlgren *Facebook Inc.* London, UK
K. Bojarczuk *Facebook Inc.* London, UK
I. Dvortsova *Facebook Inc.* London, UK
M. Harman *Facebook Inc.* London, UK
R. Hatout *Facebook Inc.* London, UK
M. Lomeli *Facebook Inc.* London, UK
E. Meijer *Facebook Inc.* London, UK
S. Sapora *Facebook Inc.* London, UK

***Abstract*—We describe Facebook's development of WW, its cyber-cyber digital twin, and, in particular, the way it uses hybrid artificial intelligence techniques for behavioural and structural imitation. WW is a simulation of Facebook user behaviour on the real Facebook platform, which Facebook uses to test and optimise its products. There are inherently many foundational scientific challenges that come to the fore when attempting to safely, reliably and effectively simulate user behaviours at scale. This paper outlines some existing practical and scientific work at Facebook, together with open problems and challenges for this research agenda, which lies at the intersection of artificial intelligence and software engineering.**

***Index Terms*—Artificial Intelligence, Software Engineering, Imitation Learning, Simulation, Multi Agent Systems, Synthetic Graph Generation, Social Media**

I. Introduction

WW is Facebook's digital twin simulation of its WWW platform [1]. WW is a web-enabled simulation [2], that simulates Facebook user community behaviour. Although it simulates users, WW executes on the real Facebook platform [1], making it highly realistic and more actionable than traditional simulation. Facebook uses WW to help test its platforms at the highest level of test abstraction; simulation-based testing assesses the impact of a change on entire communities of users. We are also exploring the use of simulation to help us find product improvements that tend to reduce the impact of users who exhibit harmful behaviour online. Ultimately, we believe that simulation can, should, and will inform the daily activity of every engineer at Facebook.

In order for simulation to scale to tackle the kinds of challenges we face at Facebook, we need extremely fast simulations, which maintain high fidelity to the real world problems they simulate. This is a scientific and engineering challenge, because increased fidelity typically requires a concomitant increase in execution time. To tackle this problem, WW deploys as a hierarchy of simulations, each of which balances the trade-off between fidelity and execution time [1]. In this paper we describe our approach to faithfully simulating both user behaviour and the community structure, outlining some of our approaches to assessing the fidelity of the simulations.

Mark Harman's scientific work is part supported by European Research Council (ERC), Advanced Fellowship grant number 741278; Evolutionary Program Improvement (EPIC) which is run out of University College London, where he is part time professor. He is a full time Research Scientist at Facebook. Author order is alphabetical.

Scientific problems associated with simulation have attracted a great deal of interest and have acquired significant importance. For example, simulation is increasingly used for prediction and decision-making in economics [3], climate change and weather prediction [4], traffic safety [5], and, most recently, with significant immediate impact, the global response to the COVID19 pandemic [6]. While some of the work reported here is specific to Facebook, we believe that much of it generalises to other simulation problems, especially those that involve modelling user behaviour, which is increasingly important to test and optimise software engineering products.

II. Mechanism design: searching the design space

We have used an approach reminiscent of genetic improvement [7] that we call 'mechanism design' [2], to explore simulated changes to Facebook products, together with their simulated impact on reducing harmful behaviour online. In our experiments we simulate both the product changes themselves, and their impact on user communities, using WW. The product changes are simulated as interventions that impact each bot's actions and observations. The impact on user communities is assessed by simulating both normal users and users that engage in harmful behaviour online.

The goal of this work is to identify product changes that simultaneously tend to reduce harmful behaviour, while not impacting normal behaviour. As can be imagined, this is naturally a multi-objective optimisation problem, in which the two objectives are conflicting. Therefore, the problem is ideal for computational search. We plan to develop and extend mechanism design to use computational search [8] over the space of product changes, guided by fitness functions that capture the impact of the product changes on normal and harmful behaviour.

Currently, we use heat maps to visualise the impact of changes denoted by parameters that define the product intervention being simulated. The heat map allows us to sweep over parameter choices showing, in a single visualisation, the impact of various choices of parameter on bots' behaviour. In each cell of the heat map, we compute the ability of normal users to achieve their normal goals, such as making and visiting friends, and of harmful users to achieve their harmful intentions. The visualisation helps to identify the sweet spot among the parameters that simultaneously inhibits the harmful behaviour while allowing the normal behaviour.

However, the heat maps also allow us to assess the fidelity of our overall simulation, as we use more abstract versions of the simulation, moving up the cyber cyber twin simulation hierarchy [1]. The base case for the hierarchy is execution on the real platform, which produces the most high fidelity simulation. As we move up the hierarchy, we increasingly trade fidelity for speed of simulation execution. However, we want to ensure that this trade-off still affords us acceptable fidelity for the simulation task in hand. Therefore, we compare results achieved for the base case with those achieved using (potentially) lower fidelity (but faster) simulations from higher up the hierarchy. We give an example in Section III-C.

III. Synthetic Graph Generation

The graph structure of the social network underlying the simulation is important because the graph's structural properties affect the fidelity of the simulation. We designed a privacy-safe social graph generation approach. Our approach generates synthetic graphs that are representative of the social network node features and friendship edges. Because the graph is synthetic, it abstracts away from details irrelevant to the simulation, while retaining those relevant details. This abstraction yields orders of magnitude faster simulation execution. Synthetic graphs are thus abstractions of the social graph that we use in simulations to speed up simulation time. Naturally, we do find ourselves necessarily trading some loss in simulation outcome fidelity for such dramatic speed ups. Overall, as explained elsewhere [1], we implemented a hierarchy of different simulations, allowing us to choose the overall trade off between fidelity and speed.

The generated graphs inherit a set of statistical properties of interest from the real social network's graph structure and user feature distributions. This approach allows us to create many possible social graph worlds with the same underlying statistical properties. Namely, it is possible to sample multiple graphs of different sizes. For example, for a true social graph sub-network of approximately one million users, we construct a reduced, but representative, graph of approximately 10,000 nodes, sampled in order to retain relevant sets of statistical properties inherited from the true sub-network.

Synthetic graphs support faster simulation, and also allow us to experiment with counterfactual scenarios in which we permute the structure of the underlying graph, or properties of its nodes. In a counterfactual world, some properties resemble those of the real world, while others are mutated. In this way we can understand the influence of coarse-grained structural properties of the social graph on behaviour and optimise products accordingly.

A. Graph Generation Algorithms

A graph consists of a set of vertices (also called nodes) which are connected by edges (also called links). Friendship relationships can be represented by undirected edges, while other types of social network interactions, such as messaging a friend or posting on a friend's timeline, are directed.

In this section we focus on generative models for undirected graphs which model the friendship relationships in a social network. There exist many algorithms to generate synthetic undirected graphs that inherit one or more properties from a given graph. We cover only that small subset of the literature on graph generators in this section, upon which we have based our current implementations. A more detailed survey of graph generation algorithms can be found elsewhere [9].

1) **Erdös-Rényi-Gilbert model [10], [11]**: A graph is constructed by connecting nodes randomly. Each edge is included in the graph with probability p, independently from every other edge.
2) **Baranasi-Albert [12]**: A scale free graph is generated using a preferential attachment mechanism. This model leads to power law edge distributions in the large n regime. Such a power law distribution resembles the coarse-grained distribution of friendships in the real Facebook social graph, which also exhibits a power law.
3) **Configuration model [13]**: Given a degree sequence, the configuration model generates a random graph. The degree sequence can be recovered from the node's degree distribution for the friendship graph, after specifying the size of the generated graph. The degree sequence allows us to encode both power law and Poisson distributions in the large n regimes, which makes this model a useful general framework for graph generation.

The majority of graph generation algorithms reproduce only the structural properties of the graph, without taking into account node-specific attributes. Node specific attributes often exhibit correlations which may be hard to replicate; the number of possible feature set combinations grows exponentially with feature vector dimension. Nevertheless, for simulation we clearly need to include these. If we do not preserve such properties, then important attributes of the social graph will not be retained and may lead to unreliable simulation outcomes. For example, in order to accurately simulate crawling behaviours, we need to take account of properties such as demographic homophilia [14], a property we test using our metamorphic testing for structure [15]. Fortunately, the literature does include work that incorporates feature correlations [16].

Most current graph generation approaches also tend to ignore the clustering present in a graph, focusing instead, on preserving other network statistics. However, the small world phenomenon is present in many real world networks [17]. Watts and Strogatz [18] have shown that social networks are characterised by a short path lengths and large clustering coefficients. These clustering properties must also be taken into account, when constructing synthetic graphs for social media simulation. Space does not permit a detailed description of our algorithm for synthetic graph generation here, but it will be made available in a subsequent publication [19], that describes our *persona-based generative graph model*.

B. Synthetic Graph Properties

For different applications of simulation, we need to construct different kinds of synthetic graph. In this section we briefly review the properties of synthetic graphs we have currently found to be useful. This list is merely indicative, and intended to illustrate the kinds of property of concern for social media simulation. As the number of WW use cases increases, we expect this list to grow.

WW synthetic graphs currently preserve many structural properties, including:

1) **Node degree distribution**: the number of connections (friends) the users have.
2) **Clustering coefficients**: This is one particular measure of community structure that has been widely used in the literature. It measures the degree of transitivity of a graph: higher values imply that 'friends of friends' are themselves likely to be friends, leading to a 'clumpy' overall graph structure. Interestingly, these clustering properties also have profound impact on other non-social graph properties. For example, program dependence clusters [20] tend to lead to large program slices [21], and have implications for bug prevalence [22]; the 'friends of friends clumpiness' property *also* applies to program graphs.
3) **Assortativity coefficient**: This is a measure of the preference to attach to nodes that share similar properties. This preference is a form of demographic homophilia [14], [15] that our synthetic graphs need to reflect.
4) **Mixing matrix**: the tendency of a node type to connect to another node type (a bot persona).
5) **Average shortest path between two nodes**: A measure of reachability, related to small world models [23].

C. Assessing Fidelity at Different Levels of the Simulation Hierarchy with Mechanism Design Heat Maps

One way to assess the fidelity of a synthetic graph is to compare the results obtained using this graph with those obtained using a more high fidelity simulation approach. We do this using mechanism design, described in Section II.

Specifically, we simulate harmful user behaviours, using techniques for imitation learning of human user behaviour described in Section IV. We use mechanism design to simulate the effect of possible product changes on this harmful behaviour, hoping to find changes that will tend to inhibit harmful behaviour online.

From different demographics of harmful behaviours, and different kinds of harmful behaviour, we obtain a heat map that depicts the impact of the possible product change on this harmful behaviour. The heat map is used to identify and recommend product changes.

By comparing the heat map obtained from a synthetic graph, with that obtained from an online graph, we also obtain a visualisation that helps us assess the *fidelity* of the synthetic graph. Figure 1 presents an example, in which we were able to construct a high fidelity synthetic graph.

With this synthetic graph we were able to explore product changes to inhibit a particular kind of harmful behaviour with two orders of magnitude greater speed of execution, while remaining very faithful to the simulation behaviour observed in the online social graph.

D. Assessing Fidelity By Comparison with Real World Data

We use real-world data to benchmark and calibrate our simulations, assessing the fidelity of all layers of the simulation hierarchy including synthetic graphs. Figure 2 gives an example of age preferential attachment in synthetic graphs we generated to respect demographic homophilia [14], using the assortativity coefficient. We can see that these results closely resemble those from the real world social graph [24].

We also compare predictions from simulations with those observed in real-world scenarios. For example, we have used WW to simulate various forms of 'infection spread' in the social graph, using information diffusion models.

This is a problem that has been widely studied in simulation frameworks. The shape of the graph is of great importance for reliable simulations of diffusion models of any kind. For instance, the lower the node degree, the slower the infection will tend to spread. We created such synthetic graphs, applied them to diffusion model simulation, and found that we were able to reproduce real diffusion behaviours, observed in practice, with high Fidelity, as Figure 3 shows.

IV. Learning and Imitating User Behaviour

For any social media simulation to be effective, it is clearly important for the bots to imitate real users' different behavioural patters. We cluster users that behave similarly to create archetypes of behaviour. We never imitate any *individual* user's behaviour, but rather imitate the gross statistical properties of whole classes of users. We refer to these behavioural archetypes as *personas*.

Through the use of personas we are able to realistically simulate the behaviour of different sub-populations interacting with each other, as well as imitating the naturally occurring variance within each group. Gathering statistics from tens of thousands of users to create our personas also helps us avoid over-fitting, which might otherwise reduce the fidelity of the simulation. Basing our models on personas, interpolated over tens of thousands of users, also ensures that our simulations are privacy safe.

Simulating different bot personas has a number of natural applications, including

- Analysis and testing of how platform and environment changes will affect different user demographics, thereby applying social testing [2] to improve the quality of our products, and the services they offer to our customers.
- Observation of the emergent properties of behaviour across graphs with different percentages of connections amongst personas. This can help us to identify and head off challenges to the integrity of the social network, and to identify possible emergent harmful behaviours we need to tackle.

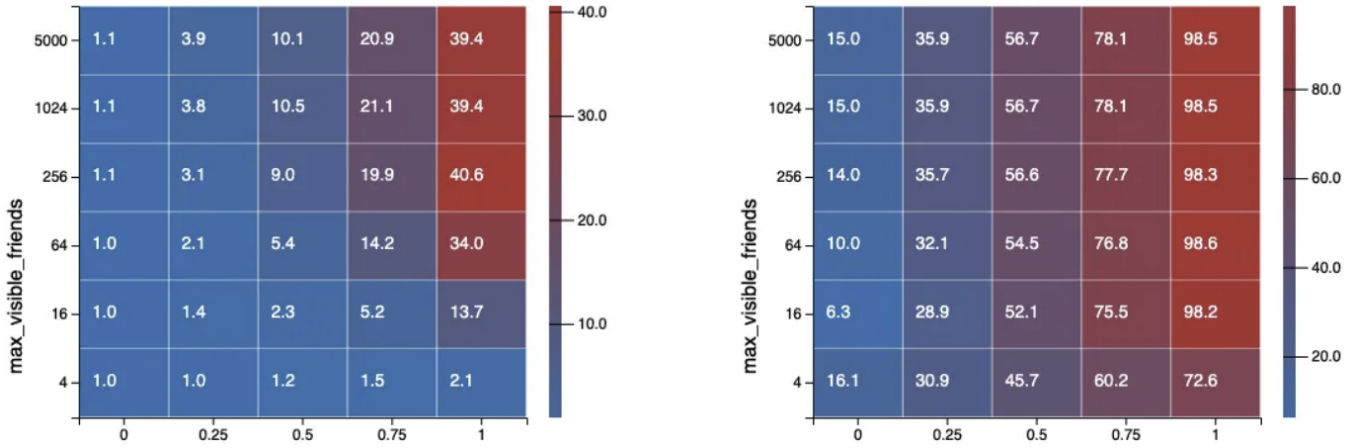

Fig. 1. Assessing the fidelity of synthetic graphs using mechanism design. Both heat maps show impact of different limits on maximum friends visible to the bot at each step (vertical axis) with different levels of propensity for the bot to engage in harmful behaviour (horizontal axis). This intervention is not in production and shown here for illustrative purposes only. The heat map on the left results from simulating on the high fidelity (but computationally expensive), online graph. The heat map on the right is result of simulating on a synthetic graph. There is a scaling factor due to the difference in the graph sizes, but the overall trends are very similar in both heat maps. In this way, we gain confidence that we retain simulation fidelity while using the (offline) synthetic graph. This fidelity confidence is important, because the synthetic graph scales up, by two orders of magnitude, the speed of exploration of product interventions.

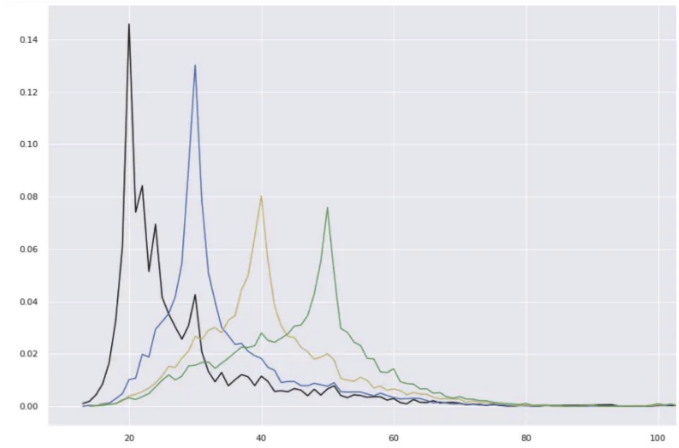

Fig. 2. The frequency distribution of ages for the neighbours (friends) of bots in a synthetic WW graph. Frequency of neighbour ages is plotted for four different age-based personas. Each of the four has peek frequency corresponding to the age demographic of the corresponding persona plotted. These distributions closely resemble those observed in the real social graph [24].

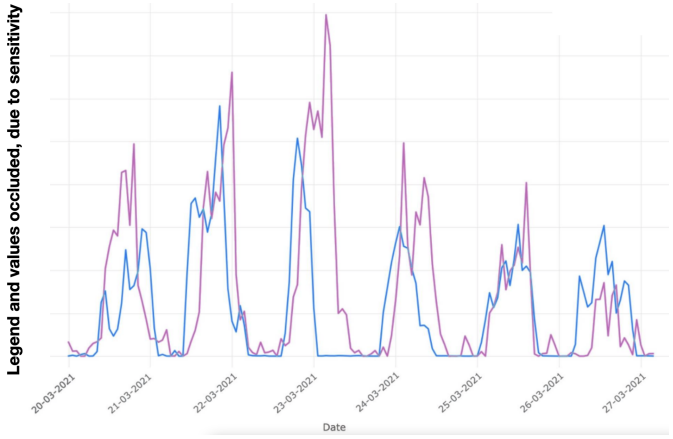

Fig. 3. Comparing real infection spread with the results of a WW synthetic graph simulation. Real data in blue (darker grey when viewed in black and white); simulated data in mauve. Although the real data is slightly left shifted, the peaks and overall periodicity are faithfully replicated in the simulation.

A. *Identifying behavioural archetypes*

We adopted several different approaches to identify personas, including

- **Demographic**: Identifying and gathering behavioural patterns based on the demographic properties of whole classes of user.
- **Unsupervised Clustering**: Behavioural patterns are embedded into a multi-dimensional space, so that k-means clustering can be used to separate the groups. Examples of groups that result from this method include groups from similar geographic regions that perform actions at similar times during the day.
- **Ad-hoc**: We also model groups of users that might be of particular interest for the exploration of a particular product improvement or intervention. For example, harmful online behaviours that we hope the product improvement will tend to reduce.

B. *Imitation Learning Through Simple Statistical Similarity*

We gather the frequencies and high-level characteristics of actions from all demographics at an hourly granularity. We use this data to build probability distributions for each action and demographic group.

Each demographic group has its own statistical model. The statistical model contains a probability distribution for each action for each hour and day, where each action frequency maps to its corresponding probability.

This level of granularity allows us to replicate behaviour with high fidelity. We can build bots that model particular kinds of demographic, taking account of seasonal and other time-based variability, and at different levels of granularity. For instance, one group of fine-grained bots could be deployed to model Japanese women in their 30s at 10AM PST, while another more coarse-grained group of bots might represent Australians on a Monday.

For WW's simplest settings, probability distributions are independent. However, this is unrealistic, because it fails to take account of joint probability distributions. The frequencies and characteristics of different actions are not independent, but are typically correlated in subtle ways. Sadly, it is computationally intractable to compute the full joint probability distribution with large numbers of actions involved, and at Facebook scale. Therefore, we need to make approximations. Once again this raises interesting scientific questions about how to assess and optimise fidelity and computational effort. Fortunately, there is a rich scientific literature on taming the inherent combinatorial explosion, both within the context of machine learning [25] and also software testing [26].

In order to identify the likely stakeholders participating in a particular action, we leverage existing machine learning classifiers. Facebook has many such classifiers that predict, for example, the likelihood of a user interacting with a certain piece of content. We re-use these classifiers to allow bot personas to accurately and realistically imitate a class of users.

C. Imitation Learning through Machine Learning

The statistical model, although sufficiently accurate for some simulations, presents two principle shortcomings:

- **Order**: The statistical model does not capture action execution order. Users might be more likely to like a post before commenting on it, yet this is not captured by a purely 'action prevalence' statistical model.
- **Environmental Clues**: Our computed probability distributions are fixed, thereby failing account for environmental factors that might influence the bots' decision making. For example, from our statistical model, we might predict that a bot has a high probability of sending a friend request. However, that likelihood is increased when friends move to the same location, or when one of them receives a recommendation. Currently, these contextual characteristics are not modelled. More work is needed to determine environment characteristics, thereby imbuing WW bots with greater context sensitivity.

To address this we also used simple Machine Learning (ML) for imitation. For the first iteration of our ML models, we treated the problem as a classification problem. The labels from which the model can choose represent the available actions that the bot can execute. To train the model, a database was created from sequences of user actions, processed so they would be cut at random intervals. This allowed our model to predict a generic 'next action' in the sequence of actions, but left it unable to predict answers to related questions, such as 'what is the most likely final action in the sequence?'.

Our cutting algorithm can be described as follows: given a sequence s of length n, a sub-sequence s' is selected where $s' = s[0:r]$ and $r = random(1, n-1)$. s' will be of length $[2, n-1]$. We then pick our action label y to be $s[r]$. This is the action we want to predict. We repeat this process for our whole data set of action sequences, ending up with many sequences of variable length.

We used a simple feed forward neural network, because this is a simple and well-understood architecture. The best performing network was the feed forward network with 3 hidden layers, 0.3 dropout [27] and batch normalization [28].

The neural network is used to predict the next action from an arbitrary sequence of previous actions. The arbitrary sequences used for training were produced using our cutting approach. In order to provide a fixed length action sequence encoding for the neural net, we use Dual-FOFE, which has recently been demonstrated to perform optimally (with two forgetting factors) [29] when transforming variable length sequences to fixed length sequences.

The feed forward networks were trained using a super convergent [30] scheduler and tested using 10-fold cross validation to assess accuracy using negative-log likelihood. This yielded an average accuracy of 56%. We also experimented with the TabNet tabular learner which raised accuracy to an average of 64%. In future work we plan to experiment with other neural net architectures.

These results compare to a baseline with random guessing of 6.25%. However, the most prevalent action is 'like', which occurs 34% of the time. Therefore, a more realistic baseline would be to always predict that the next action will be 'like'. That approach would clearly yield superior accuracy outcomes compared to random guessing, but also considerably poorer accuracy outcomes than those we obtained from all the learners with which we have, hitherto, experimented.

D. De-identification and Privacy-safe Simulation

Simulation is an inherently privacy safe technology, for companies to test and explore improvements their products. All experiments are performed in an artificial world, created specifically to avoid reliance on any real user data. In order to ensure privacy safety, we also take careful steps to de-identify users before any bot training takes place.

E. Imitation Learning through Hybrid Rule Based Approaches

We have also experimented with simple rule-based algorithms to capture particular kinds of simulated behaviour, such as crawling behaviours. One approach with which we experimented consisted of modeling crawling behaviours using a depth-first traversal of the social graph. On its own this behaviour is of little value, but when combined with simple machine learning techniques it provides a technique that offers the technical advantage of balancing interpretability and realism. We combine simple rule based behaviours with machine-learning-defined decision procedures to predict next actions, such as those outlined in Section IV-C.

Figure 1 shows the outcome of this approach in simulating classes of users by bots that exhibit harmful behaviours in the simulation. Naturally, all harmful behaviour is confined to the WW simulation environment, and involves only bots, thereby providing engineers and data scientists with a safe environment in which to experiment with product improvements that tackle these harmful behaviours.

In these WW experiments, we combined a depth first traversal algorithm with a decision procedure that was informed by simple logistic regression. At each point in the graph traversal at which the bot has a choice of candidate nodes to visit, the bot makes that choice using the logistic regression. The regression procedure trains the bot to make decisions similar to classes of users they have exhibited particular kinds of harmful behaviour.

In this way, the bot is trying to behave similarly to a whole class of users, without modelling any particular one. Furthermore, the behaviour of the bot is comparatively more explainable than the behaviour of bots trained purely with machine learning. The approach allows us to control the degree of harmful intention, by regulating how often the bot uses logistic regression trained on harmful behaviours, and how often it uses alternative approaches trained on normal behaviour.

V. Conclusion

We have outlined how we model user behaviour and social structure to support simulations in WW. We explain some of the approaches that were found useful in assessing the fidelity of the simulations. This has allowed us to execute simulations two orders of magnitude faster, without notably trading simulation fidelity. We are excited to work with the scientific community on this social media simulation research agenda. We hope this paper helps outline some of the open problems and challenges, thereby facilitating future collaboration.

References

[1] J. Ahlgren, K. Bojarczuk, S. Drossopoulou, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, S. Lucas, E. Meijer, S. Omohundro, R. Rojas, S. Sapora, J. M. Zhang, and N. Zhou, "Facebook's cyber–cyber and cyber–physical digital twins," in *25th International Conference on Evaluation and Assessment in Software Engineering (EASE 2021)*, Virtual, 2021.

[2] J. Ahlgren, M. E. Berezin, K. Bojarczuk, E. Dulskyte, I. Dvortsova, J. George, N. Gucevska, M. Harman, R. Laemmel, E. Meijer, S. Sapora, and J. Spahr-Summers, "WES: Agent-based user interaction simulation on real infrastructure," in *GI @ ICSE 2020*, S. Yoo, J. Petke, W. Weimer, and B. R. Bruce, Eds. ACM, 3 Jul. 2020, pp. 276–284, invited Keynote.

[3] S. Terzi and S. Cavalieri, "Simulation in the supply chain context: a survey," *Computers in Industry*, vol. 53, no. 1, pp. 3–16, 2004.

[4] G. L. Johnson, C. L. Hanson, S. P. Hardegree, and E. B. Ballard, "Stochastic weather simulation: Overview and analysis of two commonly used models," *Journal of Applied Meteorology*, vol. 35, no. 10, pp. 1878–1896, 1996.

[5] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *Journal of Network and Computer Applications*, vol. 37, pp. 380 – 392, 2014.

[6] D. Adam, "Special report: The simulations driving the world's response to COVID-19," *Nature*, April 2020.

[7] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward, "Genetic improvement of software: a comprehensive survey," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 415–432, Jun. 2018.

[8] M. Harman, "The current state and future of search based software engineering," in *Future of Software Engineering 2007*, L. Briand and A. Wolf, Eds. Los Alamitos, California, USA: IEEE Computer Society Press, 2007, this volume.

[9] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Comput. Surv.*, vol. 38, no. 1, p. 2–es, Jun. 2006. [Online]. Available: https://doi.org/10.1145/1132952.1132954

[10] P. Erdös and A. Rényi, "On random graphs," *Publicationes Publicationes*, vol. 6, p. 290–297, 1959.

[11] E. N. Gilbert, "Random Graphs," *The Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141 – 1144, 1959. [Online]. Available: https://doi.org/10.1214/aoms/1177706098

[12] R. Albert and A. lászló Barabási, "Statistical mechanics of complex networks," *Rev. Mod. Phys*, 2002.

[13] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Phys. Rev. E*, vol. 64, p. 026118, Jul 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.64.026118

[14] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[15] J. Ahlgren, M. E. Berezin, K. Bojarczuk, E. Dulskyte, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, E. Meijer, S. Sapora, and J. Spahr-Summers, "Testing web enabled simulation at scale using metamorphic testing," in *International Conference on Software Engineering (ICSE) Software Engineering in Practice (SEIP) track*, Virtual, 2021.

[16] J. J. Pfeiffer, S. Moreno, T. La Fond, J. Neville, and B. Gallagher, "Attributed graph models: Modeling network structure with correlated attributes," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 831–842. [Online]. Available: https://doi.org/10.1145/2566486.2567993

[17] M. E. J. Newman, "Random graphs with clustering," *Phys. Rev. Lett.*, vol. 103, p. 058701, Jul 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103.058701

[18] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, p. 440–442, 1998.

[19] J. Ahlgren, K. Bojarczuk, M. Harman, M. Lomeli, and S. Sapora, "Persona-based graph generative models," 2022 (in preparation).

[20] M. Harman, D. Binkley, K. Gallagher, N. Gold, and J. Krinke, "Dependence clusters in source code," *ACM Transactions on Programming Languages and Systems*, vol. 32, no. 1, Oct. 2009, article 1.

[21] D. Binkley and M. Harman, "Locating dependence clusters and dependence pollution," in 21^{st} *IEEE International Conference on Software Maintenance*. Los Alamitos, California, USA: IEEE Computer Society Press, 2005, pp. 177–186.

[22] Y. Yang, M. Harman, J. Krinke, S. S. Islam, D. Binkley, Y. Zhou, and B. Xu, "An empirical study on dependence clusters for effort-aware fault-proneness prediction," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016*, 2016, pp. 296–307. [Online]. Available: http://doi.acm.org/10.1145/2970276.2970353

[23] M. E. Newman, "Models of the small world," *Journal of Statistical Physics*, vol. 101, no. 3, pp. 819–841, 2000.

[24] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, "The anatomy of the Facebook social graph," *arXiv preprint arXiv:1111.4503*, 2011.

[25] S. Bengio and Y. Bengio, "Taking on the curse of dimensionality in joint distributions using neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 550–557, 2000.

[26] J. Petke, M. B. Cohen, M. Harman, and S. Yoo, "Practical combinatorial interaction testing: Empirical findings on efficiency and early fault detection," *IEEE Transactions on Software Engineering*, vol. 41, no. 9, pp. 901–924, 2015.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 1929–1958, Jan. 2014.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[29] S. Watcharawittayakul, M. Xu, and H. Jiang, "Dual fixed-size ordinally forgetting encoding (fofe) for competitive neural language models," in *EMNLP*, 2018.

[30] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2018.