Optimization of Large-Scale Agent-Based Simulations Through Automated Abstraction and Simplification

Alexey Tregubov(✉) and Jim Blythe

USC Information Sciences Institute, Marina Del Rey, CA, USA
{tregubov,blythe}@isi.edu

Abstract. Agent-based simulations of social media platforms often need to be run for many repetitions at large scale. Often, researchers must compromise between available computational resources (memory, run-time), the scale of the simulation, and the quality of its predictions.

As a step to support this process, we present a systematic exploration of simplifications of agent simulations across a number of dimensions suitable for social media studies. Simplifications explored include sub-sampling, implementing agents representing teams or groups of users, simplifying agent behavior, and simplifying the environment.

We also propose a tool that helps apply simplifications to a simulation model, and helps find simplifications that approximate the behavior of the full-scale simulation within computational resource limits.

We present experiments in two social media domains, GitHub and Twitter, using data both to design agents and to test simulation predictions against ground truth. Sub-sampling agents often provides a simple and effective strategy in these domains, particularly in combination with simplifying agent behavior, yielding up to an order of magnitude improvement in run-time with little or no loss in predictive power. Moreover, some simplifications improve performance over the full-scale simulation by removing noise.

We describe domain characteristics that may indicate the most effective simplification strategies and discuss heuristics for automatic exploration of simplifications.

Keywords: Abstractions · Simplifications · Agent-based simulation · Massive scale simulations · Online social networks

1 Motivation

Large-scale simulations may be used for many purposes, including prediction and exploration of what-if scenarios. Typically, a large number of parameters, such as behavioral characteristics of individual agents, may not be known precisely but are modeled probabilistically, requiring many iterations of the simulation as parameters are varied systematically to arrive at an estimate that is accurate

S. Swarup and B. T. R. Savarimuthu (Eds.): MABS 2020, LNAI 12316, pp. 81–93, 2021.
https://doi.org/10.1007/978-3-030-66888-4_7

and whose dependence on the parameter space is known. This can be a very expensive process.

A common solution is to iterate and test over smaller, simpler versions of the problem that are chosen to provide a close estimate to the simulation outcomes under test while requiring considerably less run-time and memory to run. In some cases, the final estimate may be more precise although each individual run may be less so, because many more iterations are possible. The simplifications used are typically drawn from a standard set of broad categories including (1) reducing the number of agents and considering a sub-region of the original simulation, (2) simplifying the agent decision processes to use less computation and memory, (3) simplifying the representation of the environment, (4) selectively replacing subsets of either the agents, the environment or both with direct models of the behavior of a group of agents or region of the environment, and (5) simplifying and/or reducing the communication that takes place between agents, which often dominates the computational complexity of a simulation. Domain-dependent simplifications, for example based on geographic constraints of agents, may combine several of these categories. Often, developers follow an ad hoc process: while the simulation may be simplified in many ways, only one or two are typically used, without evidence for which simplification may yield the best estimates of the parameters to optimize the full-scale simulation. While transformations may be required in order to adapt the results from the abstract problem into the full-scale problem, they are typically not explored in detail.

In this paper we take a first step towards an empirically-based tool for selecting an appropriate simplification from a rich set of possible approaches in a realistic domain. While other work, described below, has used formal methods to show that an abstraction will provide the correct result and is as simple as possible, the empirical approach we propose can be applied to large simulations that may require the use of code that is not sufficiently modeled to support a formal approach. We define a number of ways in which a multi-agent simulation may be simplified, and for each one we discuss the kinds of information about the full-scale simulation that may be most faithfully estimated by this simplification. We investigate a number of simplifications in the context of two domains: a massive simulation of GitHub users, with trace data describing around 10 million users and 30 million repositories, and a large simulation of Twitter containing 650 k users and 1.6 million tweets. We introduce a full-scale simulation to predict the next two months of activity on GitHub or Twitter based on four months of trace data, in which each agent is described by statistics on their past history and a set of parameters describing how future behavior may be produced based on the history. Next, we explore a number of abstractions of the full-scale simulation in order to optimize parameters for the agents, and discuss which abstractions combine the most accurate estimates with the greatest savings of computational resources. These abstractions may be used individually or in combination to reduce the computational expense of running a simulation by several orders of magnitude in some cases.

We find that, in a broad range of situations, simple domain-independent modifications such as agent sub-sampling can yield simulations that provide predictions similar to those of the full-scale simulation with a fraction of the time or memory requirements. In other cases, domain-dependent simplifications are required for such improvements, and we show how they can be derived from domain-independent principles, such as reducing the action space of the agents or aggregating agents. In a small but significant number of cases, the reduced simulation yields better prediction results than the full-scale simulation using a fraction of the resources. This happens when the agents, actions or world features removed were predominantly a source of noise for the predictions of interest.

Finally we review other large-scale simulations from the literature to verify that the abstractions we consider make sense within those simulations and may be expected to yield useful results. We also discuss how the framework for abstractions may be used to run simplified simulations in cases where the full-scale simulation is infeasible, and how experiments such as the ones we describe may help indicate the most useful simplifications within a single partially abstracted simulation. This is an active area of investigation within our group.

The novel contributions of this paper include the first comparative, empirical exploration of a broad set of simplification criteria across multiple simulation domains. While previous work, described below, investigated one or more simplifications in a simulation domain, we compare an expanded set of techniques within the same framework and discuss the extent to which they are domain-independent or whether domain dependent modifications may be necessary to apply them. Another contribution is an initial framework to help experimenters to consider a broader set of possible simplifications that lays the groundwork for semi-automated tools to simplify simulations.

2 Related Work

Struss [15] distinguishes abstraction, simplification and approximation in the context of model-based reasoning. In this view, abstraction and approximation are both special cases of simplification, which is a general change to a model, perhaps altering relations between variables describing a problem, that reduces some modeling details. An abstraction is a kind of simplification that proposes a new model, e.g. by changing the set of variables or their domains. An approximation may replace a function in the model with a simpler version, some of whose values differ from the original. Our work broadly follows this categorization. De Kleer [6] and others address the question of finding the appropriate level of abstraction for a problem, defined as the simplest abstract version that is adequate to solve the problem. Here we explore the same issue, but empirically rather than analytically, since elements of a simulation may take the form of complex code for which such reasoning is intractable. In this paper, we use term simplification for all abstractions, simplifications and approximations.

Work on abstractions in planning systems may be viewed as simplifications to the decision space of a single agent or its environment, e.g. [9,10]. This work

motivated us to explore reductions in the action space for agents in general simulations. In a similar way, Cohen et al. derive abstractions in multi-agent simulation systems that provably preserve certain temporal properties of the simulation [5].

Simplifications and abstractions were explored in simulation of biochemical systems by Rhodes et al. [13]. To reduce model complexity Rhodes et al. experimented with scale (number of agents), time step, complexity of inter-agent messaging and conflict resolution. Authors used runtime and accuracy as metrics to evaluate the impact of each simplification. They conclude that some simplifications (e.g. number of agents, time step) can be varied significantly to reduce runtime within the reasonable accuracy range. In this paper, in addition to runtime we also explore memory usage and various accuracy metrics for social networks.

Shirazi et al. [14] dynamically replace groups of agents that occupy the same geographical area with a single agent representing the group. This is a spatial version of the general principle of aggregating agents based on shared interaction, which we define below and present a social network-based approach for in Sect. 4.3.

There is a comprehensive body of research on developing domain and application specific abstractions and simplifications [3,8,12,13]. However, finding simplifications and identifying ranges of parameters that produce useful reductions is often a manual process. Automated search is rarely discussed in literature.

3 Simplification Types

In this section we discuss a number of general types of simplification, shown in Table 1. Some of these represent domain-independent simplification methods that can be used in any multi-agent simulation while others are domain-independent principles that may or may not yield domain-dependent methods for a given problem. As we discuss the trade-off of computational resources consumed and accuracy below, we assume that there is some metric that is applied to the simulation, for example it might be used to evaluate which is best of a set of potential policies, or make predictions given some initial conditions. In the absence of ground truth for the metric, we seek a simplification that performs as closely as possible to the full-scale simulation on the metric, although it may behave quite differently on measures that are not of interest to the problem.

3.1 Subsampling

Simulation of the entire population can be computationally expensive. Subsampling identifies a smaller subset of agents and resources with the aim of reproducing the behavior of the entire population in order to meet the original goal of the simulation. Identifying a smaller subset of agents and resources that can reproduce behavior of the entire population may require exploring a wide range of parameters. One approach is to exploit any structure that might be found within the simulation framework, for example:

Table 1. Types of simplifications. Rows correspond to aspects of the simulation to be simplified, while columns denote whether the simplification is a reduction of a group of objects or involves creating a new model.

Target	Reduction	Abstraction
Set of agents	E.g.: Subsampling of agents	E.g.: Aggregate agents into meta-agents
Agents' decision space	E.g.: Reduce action types	E.g.: Aggregate actions
Environment	E.g.: Reduce resources used	E.g.: Aggregate resources,environment data
Set of events	E.g.: Reduce amount of generated events	E.g.: Aggregate events
Agents' communication	E.g.: Frequency of synchronization among agents	

- When simulating a social network one may select a smaller number of agents and associated resources by selecting a small number of connected components.
- For simulations where historical data on agents' activity is available, one can select agents that were active recently, disregarding agents which were not active after some threshold.
- If the environment allows geographical segmentation, running the simulation on a smaller number of segments can reduce the number of agents while preserving local properties.

This class of simplifications may be one of the easiest to apply in a domain-independent way, since it is possible to treat agents and their decisions and communications as black boxes. For example, domain-independent strategies for subsampling include random selection. In this paper we discuss random and frequency-based subsampling of agents, resources and actions/events.

3.2 Simplifying Agents' Decision Process

Each agent's decision process may be complex, possibly depending on many environmental conditions, leading to a heavy computational burden. To reduce runtime we can simplify agents' decision making process. Agents' behavior simplification can be applied to all agents or to just a subset. Making a small simplification in the decision process in a simulation with millions of agents can significantly reduce overall runtime.

There are different ways to approach agents' decision process simplification:

- Approximation of data and parameters needed to make decision. For example, decision process can be approximated with probabilistic models using historical data. This could be precomputed in advance.
- Simplification of decision rules, reducing the number of steps in the process.
- Reducing possible actions in agent's decision process.

Additionally, depending on the process this approach can reduce size of the agent and overall memory requirements. For example, if aggregate measurements and values are no longer needed this memory can be released.

Simpler decision processes tend to require less external data, which makes agents less dependent on the environment. This property is useful when parallel simulation is developed, it simplifies synchronization of the shared environment.

This approach has limitations. Just as small simplifications to each of millions of agents may have a significant impact on runtime and memory usage, small discrepancies in the behavior of each agent may combine to produce significant inaccuracies in metrics applied to the simulation.

This simplification is often domain-dependent since it modifies internal processes of the agent.

3.3 Selectively Replacing Groups of Agents with One Agent

In a large-scale simulation it is often possible to identify classes of agents that share similar properties and behavior. Some of these classes can be modeled as one agent that represents activities of the group as a whole, effectively creating a simulation that combines components modeled at different resolutions.

It is possible to apply simplifications of this type in a domain-independent manner since external features of the agents, such as geographic location or agent type, can be used to form groups while treating the agent as a black box and aggregating the observed actions. However more effective groups tend to be found using domain-dependent methods that can exploit regularities in decision-making, for example, or allow communication to be localized.

3.4 Simplifying Communication Between Agents

In large-scale simulations communication can consume a lot of computational and network capacity. In distributed parallel simulations, this problem is especially acute because of the network limitations.

We note that this is a distinct strategy from optimizing communication in the simulation. In optimization, the same information is shared between agents at the same time as in the original simulation. In simplification, communications may be degraded in content or timing, trading off fidelity of the simulation for memory and run-time reductions. Methods for optimization include caching frequently sent information and using communication hubs to decentralize information distribution. In distributed parallel simulations, one may partition the agents across computational nodes so as to minimize expensive cross-node communications [1].

There are different approaches to simplifying communication among agents:

- Reduced/simplified communication, where messages may be strategically dropped or summarized.
- Bulk information update, in which eventually the same information is communicated, but at any given time an agent may have received less information than in a high-resolution communication environment.

Other simplifications can also help to reduce resource consumption (e.g. memory) and allow researchers to use fewer compute nodes or in some cases avoid parallel computations altogether. Reducing the number of communicating compute nodes reduces communication overhead among agents and the environment. This simplification is often used in combination with others.

This simplification is often domain-dependent, because it requires knowledge of the environment and agents' communication protocols.

4 Experiments

4.1 Simplifications Support in FARM

FARM is an agent-based simulation framework with support for large-scale simulations. It is implemented in Python. FARM architecture provides components to model social networks [1–3]. We conducted our experiments with simplifications using FARM. FARM supports some of the simplification operators for simulation models described above. It also implements a simple search for operator parameters that yield a compromise between runtime and memory and a quality metric of the simulation.

4.2 GitHub and Twitter Simulations

GitHub is a hosting platform for software repositories using the git version control protocol, that also provides additional features such as wikis, issue-tracking, discussion boards. GitHub is an example of a social network where users can comment on commits, fork repositories, create branches, make pull requests etc.

We developed a multi-agent GitHub and Twitter simulations using DASH agents [4,11]. In our GitHub simulation model DASH communication hubs provide access to repositories and other shared state information. DASH agents perform action on repositories (e.g. push to a repository, make a pull request, etc.). Agents' decision process is based on past history of interactions with repositories which could is obtained from historical training data. If historical data on interactions is not available agents use generalized model to choose action and repository. Frequency of actions is obtained from historical training data as well.

In our Twitter simulation model DASH communication hubs provide access to popular tweets and conversation threads. DASH agents perform actions such as tweet, retweet, quote, reply, etc.

4.3 Experiment Setup

The following simplification operators were applied to the GitHub simulation:

Random Subsamples of User Agents. The following sample sizes were used: 0.1 M, 0.4 M, 0.8 M, 1 M, 1.2 M, 1.4 M, 1.8 M. One month of training data contains 1.9 M users.

Random subsamples of repositories. The following sample sizes were used: 0.1 M, 0.2 M, 0.4 M, 0.8 M, 1.6 M, 2.4 M. One month of training data contains 2.8 M repositories.

Random subsamples of training events. The following sample sizes were used: 10 K, 0.1 M, 1 M, 10 M, 20 M. The total number of events in the training data (1 month) was 31 M.

Different amount of training data – 1d, 2d, 4d, 1 week, 2 weeks, 1 month, 2 months of training intervals were used. This simplification picks a chronological window and keeps only the agents, repositories and events that appear in that window.

Simplifying agent's behavior by reducing the number of possible event types users can produce.

Random subsamples of training events and different event types combined together. This is simplification applies to operators: Random subsamples of training events (10 K, 0.1 M, 1 M, 10 M, 20 M events) and simplification of agents' behavior by reducing the number of possible event types users can produce (only half of the most frequent event types was used).

Frequency-based subsampling of user agents. Only agents with the highest rates of actions were selected. We used the following subsample sizes: 0.1 M, 0.4 M, 0.8 M, 1 M, 1.2 M

Frequency-based subsampling of user repositories. Only agents that interact with repositories that have highest rates of actions on them were selected. We used the following subsample sizes: 0.1 M, 0.2 M, 0.4 M, 0.8 M, 1.6 M

Push star agents is a simplification of agent actions. It was observed that some users tend to produce long sequences of action (in this case it was push to a repository) that are repeated over time. Push star agents instead of making each push individually produce batched of such actions, which should potentially reduce time on handling each action individually.

Agents aggregated into team/group agents. We ran experiments with 100 K, 400 K, 700 K users grouped into teams. Teams are defined as sub-clusters (sub-graphs) of users that interact with shared set of repositories. Team agent is a simplification that aggregates properties of all users of the team and interacts with repositories of this team.

The following simplification operators were applied to Twitter simulation:

Random subsamples of user agents. The following sample sizes were used: 10 K, 50 K, 0.1 M, 0.2 M, 0.4 M, 0.6 M. One month of training data contains 0.65 M users.

Random subsamples of tweets. The following sample sizes were used: 0.1 M, 0.2 M, 0.4 M, 0.8 M, 1.6 M. One month of training data contains 1.6 M events.

Random subsamples of training events (tweets, retweets, quotes, replies, etc.). The following sample sizes were used: 0.1 M, 0.2 M, 0.4 M, 0.8 M, 1.6 M.

Different amount of training data – 1d, 2d, 4d, 1 week, 2 weeks, 1 month, 2 months of training intervals were used.

Frequency-based subsampling of user agents. Only agents with the highest rates of actions were selected. We used the following subsample sizes: 0.1 M, 0.2 M, 0.3 M, 0.4 M, 0.6 M.

These simplifications are not mutually exclusive. In many instances they are applied together. We experimented with random subsamples of training events applied together with reducing the actions considered by each agent. Another example is the different amount of training operator, which can be applied together with any simplification operator.

One month of training data with all users and resources (repositories, tweets, etc.) is considered a full-scale simulation, although it is also just one possible value of the operator.

5 Results

To evaluate quality of predictions of our simulation models we used we use the following metrics for the GitHub simulation:

User popularity - top 5000 most popular users, popularity measured as the total number of *watch* and *fork* events on repositories owned by user. Calculated as Rank-Biased Overlap between ground truth and simulation (RBO) [16].

Community Contributing users - the proportion of users who interact with a community and who are active contributors, making commits and pull requests to community repositories. Calculated as absolute difference between ground truth and simulation.

User activity distribution - the distribution of the number of events produced by users. Calculated as Jensen-Shannon (JS) divergence [7].

For Twitter simulation we used the following metrics:

User activity distribution - the distribution of the number of events produced by users. Calculated as JS divergence (ground truth vs. simulation).

Most active users - the top 5000 users with the most events. Calculated as Rank-Biased Overlap between ground truth and simulation (RBO) [16].

Every point in each figure represents a simplification operator applied to the original simulation model. Each point corresponds to a specific configuration parameter of the operator. For example, subsample size is a configuration parameter of random user, repository, event subsampling operators; the number of supported events is a parameters of the reduction of event types generated by agent. Each point is an average of 7 runs, confidence intervals are plotted on both axes.

Ranges of operators' parameters were selected manually. Automated search may allow to find suboptimal settings with fine granularity for a given metric and resource (memory and runtime) constraints.

For all metrics higher is better, 1 is the best value (perfect prediction), 0 is the lowest possible value. Figures 1 a, b show GitHub simulation evaluation metrics, runtime and memory used. As a baseline for comparison we chose simulation that uses one month of training and instantiates all agents and uses all resources from training data. For GitHub it is 1.9 M users and 3.2 M repositories, baseline simulation consumes 24.5 Gb of memory and 26 min of runtime. For Twitter it is 650 K users and 1.6 M tweets, baseline simulation consumes 7.8 Gb of memory and 172 min of runtime.

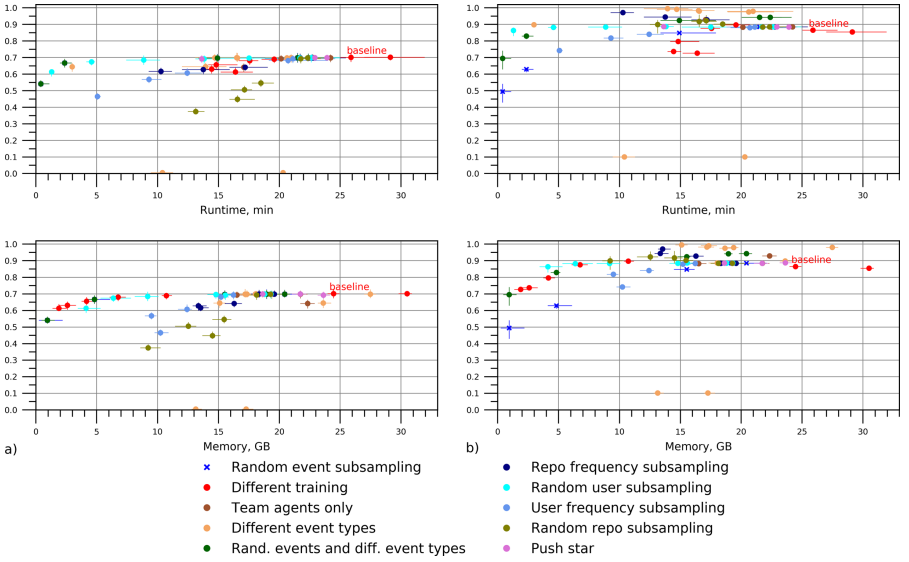

Fig. 1. GitHub: a) User popularity, b) Community contributing users

In Fig. 1a, the random event subsampling shows the same results as random user subsampling and performs better on most of the data points. The random event subsampling operator reduces simulation runtime by 48% compare to random user subsampling and different amount of training operators. Applying this operator allows almost twice as many iterations as the full-scale simulation with perfect performance, or six times as many with 97% of the original score.

In Fig. 1b reducing the types of actions that GitHub agents perform reduces noise and produces better scores than the full-scale simulation. Most subsampling operators converge on values close to 0.9 (10% difference between ground truth and simulation) where the reduced number of events operator shows values close to 0.98. At the same time it also reduces runtime and memory use in half compared to baseline.

Applying both random event subsampling and reducing the number of event types simultaneously improved performance on the community contributing users metric. This means that combining different operators can potentially produce better results than individual operators applied separately.

In the Twitter simulation both metrics (most active users - Fig. 2a, user activity distribution - Fig. 2b most of the simplification operators significantly reduce runtime and memory. For example, the frequency-based user subsampling that takes top 300 K users (about 50% of the whole data set) reduces runtime from 172 min to 43 min and memory from 7.8 Gb to 5.3 Gb. Random tweet subsampling reduces runtime to 20 min and memory use to 6 Gb on 100K tweet sample.

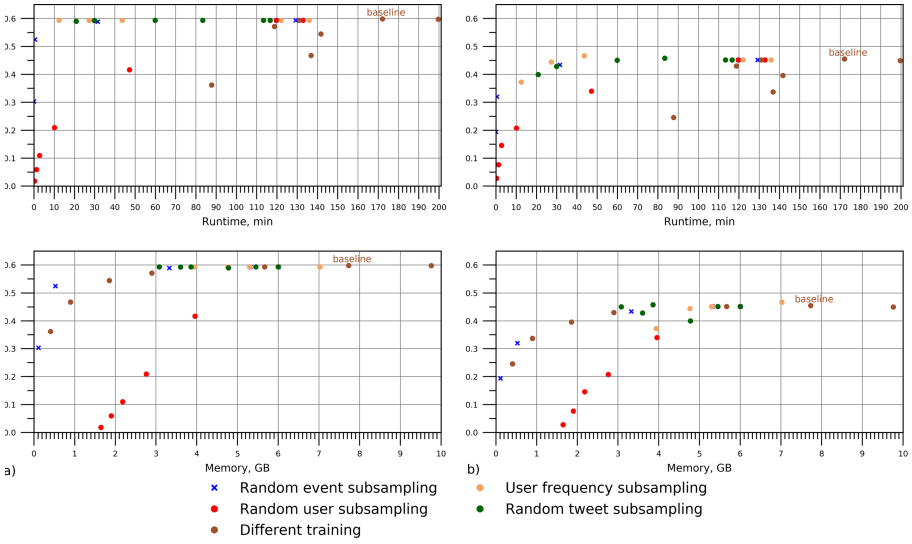

Fig. 2. Twitter: a) Most active users, b) User activity distribution

6 Semi-automated Search for Effective Simplifications

As can be seen from the previous section, some simplifications may have a dramatic effect on runtime and memory consumption for some metrics while maintaining high performance. However the best simplification is problem-dependent: random user subsampling predicts user popularity on GitHub very well in a small

fraction of the time, for example, but performs poorly on both metrics for Twitter, where frequency-based subsampling shines. Removing the right subset of actions in GitHub can reduce errors in predicting community contributing users by over 90% while running in half the time.

Finding a set of simplifications, their configurations and combinations for a simulation model is an optimization problem where the objective function balances the quality of simulation results and resource usage. We propose a tool that partially automates this process and allows researchers to find simplification configurations that yield the best results within given resource constraints. Each simplification can be viewed as a parameterized operator that is applied to the original simulation. The developer may supply domain-dependent simplification operators that make use of domain features.

If an operator's parameter space has an ordering, it is possible to use gradient descent optimization algorithms to find optimal configurations for parameters, or to use various heuristics to traverse the parameter space in search of suboptimal solution. For example, a subsampling simplification can be used with binary search on the proportion of agents to keep. This approach complements analytic approaches of e.g. [6] since it is empirical, relying on observed performance of simplified simulations. Accretion and removal operators can be used to search for the optimal set of actions for a simplified agent to consider.

7 Conclusions and Future Work

Our experiments with simplifications show that it is possible to reduce computational complexity (memory and runtime) of the simulation and preserve accuracy of the simulation. Simplifications can use domain agnostic and domain dependent algorithms. They may target specific quality metrics and properties of the simulation model. Simplifications may be applied in combination with each other.

In our experiments we demonstrated several instances of simplifications with different quality metrics on simulation of two social networks - GitHub and Twitter. In several cases simplifications reduced runtime of Twitter simulation by 85% and memory by 32%. In GitHub simulations simplifying agents' behavior by reducing the number of supported actions/events reduced error in simulation predictions to the ground truth by 90%. These findings show that applying simplifications can be useful if computational resources are constrained and simulation models requires many runs.

Possible combinations of simplifications (possible parameter values) as well as their configurations create a parameter space. The process of finding simplification settings that fit resource constraints can also be automated. Our preliminary experiments with automated search for parameters under specified runtime and memory constraints show that it is feasible to automatically identify good configurations and simplification parameters. In the future work we will explore various strategies for simplifications and their parameters. We will also add the capability for our search tool to propose and apply combinations of simplification operators dynamically.

Acknowledgements. The authors thank the Defense Advanced Research Projects Agency (DARPA), contract W911NF-17-C-0094, for their support.

References

1. Blythe, J., Tregubov, A.: FARM: architecture for distributed agent-based social simulations. In: Lin, D., Ishida, T., Zambonelli, F., Noda, I. (eds.) MMAS 2018. LNCS (LNAI), vol. 11422, pp. 96–107. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20937-7_7
2. Blythe, J., et al.: The darpa socialsim challenge: massive multi-agent simulations of the github ecosystem. In: Proceedings of AAMAS, pp. 1835–1837 (2019)
3. Blythe, J., et al.: Massive multi-agent data-driven simulations of the github ecosystem. In: Demazeau, Y., Matson, E., Corchado, J.M., De la Prieta, F. (eds.) PAAMS 2019. LNCS (LNAI), vol. 11523, pp. 3–15. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-24209-1_1
4. Blythe, J., Tregubov, A.: DASH website (2020). https://dash-agents.github.io/
5. Cohen, M., Dam, M., Lomuscio, A., Russo, F.: Abstraction in model checking multi-agent systems. In: Proceedings of AAMAS, pp. 945–952 (2009)
6. Kleer, J.: Dynamic domain abstraction through meta-diagnosis. In: Miguel, I., Ruml, W. (eds.) SARA 2007. LNCS (LNAI), vol. 4612, pp. 109–123. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73580-9_11
7. DeDeo, S., Hawkins, R., Klingenstein, S., Hitchcock, T.: Bootstrap methods for the empirical study of decision-making and information flows in social systems. Entropy **15**(12), 2246–2276 (2013). https://doi.org/10.3390/e15062246
8. Edmonds, B., Moss, S.: From KISS to KIDS – *An 'Anti-simplistic' Modelling Approach*. In: Davidsson, P., Logan, B., Takadama, K. (eds.) MABS 2004. LNCS (LNAI), vol. 3415, pp. 130–144. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32243-6_11
9. Giunchiglia, F., Walsh, T.: A theory of abstraction. Artif. Intell. **57**(2–3), 323–389 (1992)
10. Knoblock, C.A.: Automatically generating abstractions for planning. Artif. Intell. **68**(2), 243–302 (1994)
11. Murić, G., et al.: The darpa socialsim challenge: cross-platform multi-agent simulations. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS 2020 (2020)
12. Onggo, B.S., Karpat, O.: Agent-based conceptual model representation using BPMN. In: Proceedings of the Winter Simulation Conference, pp. 671–682 (2011)
13. Rhodes, D.M., Holcombe, M., Qwarnstrom, E.E.: Reducing complexity in an agent based reaction model-benefits and limitations of simplifications in relation to run time and system level output. Biosystems **147**, 21–27 (2016)
14. Shirazi, A.S., Davison, T., von Mammen, S., Denzinger, J., Jacob, C.: Adaptive agent abstractions to speed up spatial agent-based simulations. Simul. Model. Pract. Theor. **40**, 144–160 (2014)
15. Struss, P.: A theory of model simplification and abstraction for diagnosis. In: Proceedings of 5th International Workshop on Qualitative Reasoning, pp. 25–57 (1991)
16. Webber, W., Moffat, A., Zobel, J.: A similarity measure for indefinite rankings. ACM Trans. Inf. Syst. **28**(4), 1–38 (2010)