Supplementary Material for SASCA: Scalable Agent-based Simulator for Citation Analysis

Minhyuk Park[1], João AC Lamy[2], Esther CC Rodrigues[2], Felipe M Ferreira[2], Tandy Warnow[1], and George Chacko[1]

[1] Siebel School of Computing and Data Science, University of Illinois Urbana-Champaign, Urbana, IL 61801
George Chacko chackoge@illinois.edu,
https://siebelschool.illinois.edu/
[2] Insper Instituto de Ensino e Pesquisa, São Paulo,
Brazil

1 Score computation

Each node in the network has three essential attributes which are its in_degree, publication year, and fitness values. Using these attributes, their scores with respect to preferential attachment, recency, and fitness are computed as follows based on the phenotype of each citing agent.

Preferential Attachment: Preferential attachment scores, or PA for short, are given by $\texttt{max}(deg_{in}, 1)^{\gamma} + C$ where deg_{in} is the indegree of a node, γ is the exponent parameter that we set to 3, and C is the constant parameter that we set to 1. This score is then normalized to sum to 1 over all nodes.

Fitness: Fitness scores, or FIT for short, can be computed with the same formula as PA but using fitness values instead of indegrees as shown here $\texttt{max}(\texttt{fitness}, 1)^{\gamma} + C$ where `fitness` is the fitness value of a node, and scores are normalized to sum to 1 across the nodes.

Recency: Recency scores, or REC for short, are computed primarily based on the input recency table so we describe the input recency table here first. One of the required inputs to the ABM simulator is a recency table which encodes empirically derived values about real-world citation behavior. This table is created by taking a real-world network and calculating the count of edges that cite a publication from x years ago. These counts can be recorded into the table as row entries where each row has two values x and c where c is the count of edges from the real-world network that represent a citation across x years. Listing 1.1 shows an example of what a recency table might look like. In this example, the real-world citation network had $100 + 50 + 10 + 1 = 161$ edges from which 100 edges represented citations between publications published one year apart.

```
x,c
1,100
2,50
3,10
4,1
```

Listing 1.1. Example recency table

The first operation in the REC computation is the transformation of this table into probabilities. This process is a simple operation in which each count c is divided by the sum of all c values. However, one thing to note is that this is done only for valid years (i.e., over the set of unique values computed by subtracting node publication year from the current year). For example, using the example recency table, if the current year is 2000 and the only other nodes in the network were published in either 1999 or 1996, then only the c values 100 and 1 chosen and normalized in the probabilities $\frac{100}{100+1}$ and $\frac{1}{100+1}$.

The way to interpret these probabilities is that given the current network, there is a $\frac{100}{100+1}$ chance of citing a publication from 1999 (i.e., 1 year ago), and a $\frac{1}{100+1}$ chance of citing a publication from 1996 (i.e., 4 years ago). A key property that is maintained still is that the probabilities sum to 1 over all possible years. These probabilities are then split evenly across the applicable nodes. For example, if there are x nodes in the network published in 1999 and y nodes published in 1996, then each node published in 1999 will get a score that is $\frac{1}{x} \cdot \frac{100}{100+1}$ and each node node published in 1996 will get a score that is $\frac{1}{y} \cdot \frac{1}{100+1}$. This process ensures that the originally computed probabilities of citing a publication from x years ago does not become biased by the number of publications from x years ago. This process also ensures that the recency scores over all possible nodes currently in the network sum to 1.

Once each of the probabilities are divided evenly across all the nodes from the respective years, the REC computation is complete. Each node in the network has a recency score based on input empirical data, and moreover the scores all sum to 1.

Combining the scores: Each citing agent has three corresponding attributes which are weights for preferential attachment, recency, and fitness weights denoted by w_{pa}, w_{rec}, and w_{fit}, respectively. These weights are used to combine the three scores through a linear combination, yielding $\texttt{Score} = w_{pa}PA + w_{rec}REC + w_{fit}FIT$. Thus, the phenotype of each agent determines how it scores the nodes, so that two agents with different phenotypes can view the same network differently.

2 Software Commands and Versions

PyABM:

- Code location: https://github.com/illinois-or-research-analytics/abm_citations/tree/v3-dev
- Commit: ae0aa1eeb09eefd199f58fd8afd4a98ea2333f26

```
export NUMBA_NUM_THREADS=<num threads>
python abm_citations/generators/batch/batch_run_gilbertian_model.py --config ./
    base_config
```

Listing 1.2. PyABM command

```
[Environment]
cluster_data=<csv file with first two columns listing node id and publication year
    >
edge_list=<csv file with (source,target) formatted edgelist>
gamma=3
c=1
growth_rate=<growth rate e.g., 0.03 for 3\%>
num_years=<number of cycles e.g., 30>
out_folder=<output folder>
recency_table=<two column csv with (n,k) showing the number of citations (k)
    citing a publication from n years ago, derived from a real-world network>
reference_count_table=<two column csv with the second column being the possible
    out-degrees for each agent, derived from a real-world network>
same_year=<proportion of nodes citing the same year e.g., 0.12 for 12\%>

[Agent]
no_superstars=true
preferential_weight=<"random"or a floating point number indicating the weight>
recency_weight=<"random"or a floating point number indicating the weight>
fitness_weight=<"random"or a floating point number indicating the weight>
alpha=<"random" or a floating point number>
```

Listing 1.3. base_config file example

SASCA:

- Code location: https://github.com/illinois-or-research-analytics/SASCA
- Commit: 655368ebb30ece7d528be7f9c1364721739a9d01

```
abm --edgelist <INPUT_EDGELIST> --nodelist <INPUT_NODELIST> --out-degree-bag <
    OUTDEGREE_BAG> --recency-probabilities <RECENCY_PROBABILITIES> --growth-rate <
    GROWTH_RATE> --fully-random-citations <FULLY_RANDOM_CITATIONS> --num-cycles <
    NUM_CYCLES> --same-year-proportion <SAME_YEAR_PROPORTION> --output-file <
    OUTPUT_FILE> --auxiliary-information-file <OUTPUT_AUX> --log-file <OUTPUT_LOG>
```

```
--num-processors <NUM_THREADS> --log-level <LOG_LEVEL> --neighborhood-sample <
Number of nodes to sample from each neighborhood. -1 for no sampling> --
preferential-weight <-1 for random or otherwise a floating point number
indicating the weight> --recency-weight <-1 for random or otherwise a floating
point number indicating the weight> --fitness-weight <-1 for random or
otherwise a floating point number indicating the weight> --alpha <-1 for random
 or otherwise a floating point number>
```

Listing 1.4. SASCA command

3 Out_degree distribution sampled from PubMed

To better approximate realistic citation patterns, we collected a random sample of PubMed articles to empirically estimate reference out-degrees. A sequence of PubMed IDs (PMIDs) was generated by identifying the range between a randomly selected article from 2020 and another from 2025. From this range, 10,000 PMIDs were randomly sampled from a total of 8,584,929 possible PMIDs and their corresponding XML records were retrieved using Biopython's Entrez API. Each XML record was parsed to count the number of references cited by the article, either all or restricted to ArticleId IdType="pubmed", which indicates links to other articles in the PubMed dataset. In retrieving articles for our random sample, we filtered out any articles where the PublicationType was not Journal Article. Records that lacked this label or did not contain a valid reference list were excluded from analysis. Records with less than 10 references or more than 249 were discarded. This procedure was repeated across five independent runs of 10,000 articles each, resulting in a total sample of 50,000 articles. The output consisted of CSV files containing each article's PMID and its associated out-degree (reference count), which were then used to construct an empirical citation distribution to use as input. The final distribution consisted of 46,031 out_degree values ranging from 10 to 249 with first quartile, median, and third quartile values of 23, 35, and 52 respectively.

4 Community Structure

To understand the impact on community structure, we used SASCA-s to simulate the growth of two different kinds of networks with different tendencies for local citations by using two extreme α values (0.01 and 0.99). We clustered these two networks with the Leiden algorithm optimizing for CPM criterion with three different resolution values: 0.1, 0.01, and 0.001. As shown in Figure S3, at each resolution values, the clusterings of the networks where $\alpha = 0.99$ resulted in lower conductance and higher modularity, indicating a stronger community structure. However, we did notice a slight decrease in the normalized connectivity values which are computed by dividing the minimum edge cut size of each cluster by the log base 10 of the cluster size. Table S1 shows the cluster sizes for each resolution value where we can see that the median cluster sizes for the $\alpha = 0.99$ setting are almost double the median cluster sizes for $\alpha = 0.01$, which may help explain the lower normalized connectivity values for $\alpha = 0.99$ since the denominator in general would be larger.

res value	α	min	median	max	node_cov
0.1	0.01	2	5	360	0.99
0.1	0.99	2	9	435	0.92
0.01	0.01	2	19	3790	1.00
0.01	0.99	2	32	3102	0.96
0.001	0.01	2	91	20146	1.00
0.001	0.99	2	188	18395	0.99

Table S1. Cluster sizes and node coverage of SASCA-s networks clustered using Leiden-CPM SASCA-s networks were generated over 30 years and at 3% growth using the *sj* as seed with α set to either 0.99 or 0.01. Each network has 1,193,102 nodes and was clustered with the Leiden algorithm [1] optimizing the Constant Potts model at three different resolution values. From left to right, the columns indicate the resolution value for CPM-based clustering, the value of α, the minimum, median, and maximum cluster sizes, and then the node coverage (fraction of nodes in clusters of size at least two). While node coverage (the fraction of the network in clusters of at least size 2) is consistently high, cluster size increases with α and as resolution value is decreased.

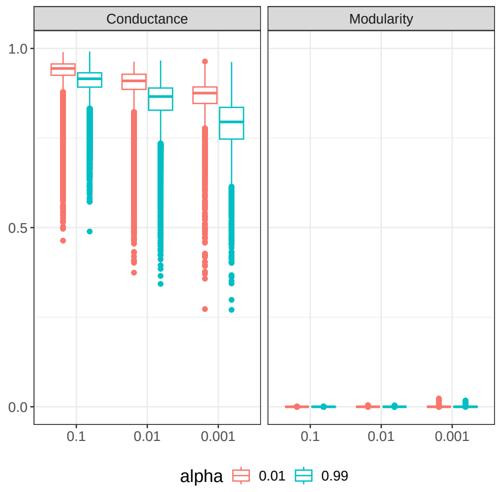

Fig. S1. Cluster quality of SASCA-s networks improves as resolution value decreases and α increases. Conductance and modularity values are summarized for non-singleton clusters from Leiden-CPM clusterings at three different resolution values (0.1, 0.01, and 0.001) for SASCA-s networks generated by 30 year 3% growth rate simulations seeded with the *sj*network using randomized agent phenotypes. While modularity scores do improve with resolution value, they remain at the lower end of the scale. Within each panel, the x-axis shows the resolution value.

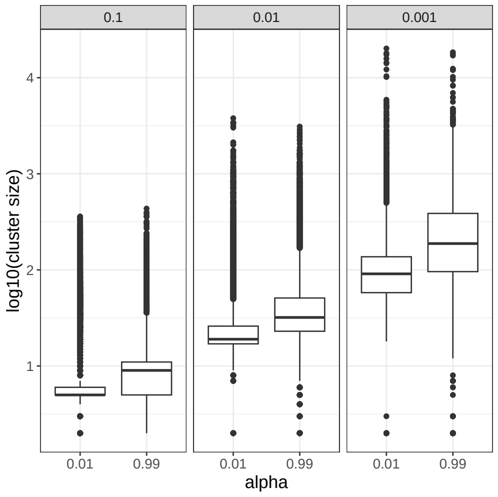

Fig. S2. Clustering statistics of SASCA-s networks. Cluster size distributions are shown for non-singleton clusters in Leiden-CPM clusterings at three different resolution values (0.1, 0.01, and 0.001) for SASCA-s networks generated by 30 year 3% growth rate simulations seeded with the *sj* network using randomized agent phenotypes. Within each subfigure, the x-axis shows the resolution value. Note that the y:axis shows log10 values for cluster sizes.

5 Seeding with Benchmarks

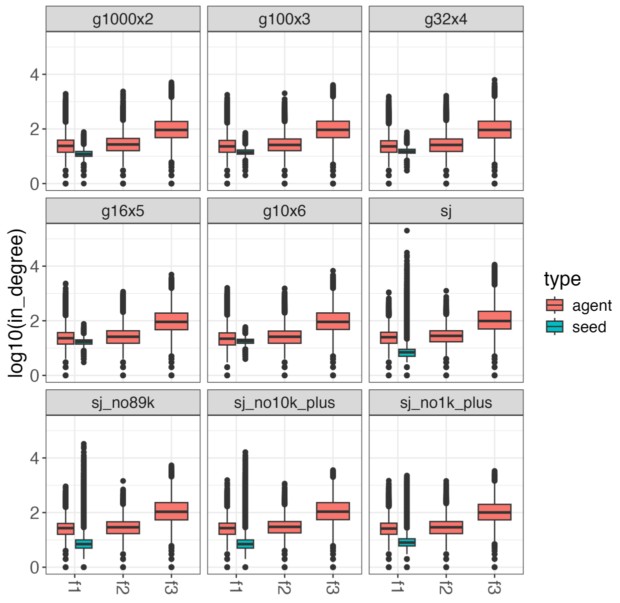

Fig. S3. Benchmark simulations. 30 yr, 3% simulations were run using a randomized agent background and SASCA-s. As seeds, lattice graphs of varying dimensions (2D-6D) and roughly a million nodes each were used as seeds (graphs labeled *g*x**) and compared with *sj* or variants of *sj* with its highest in_degree node depleted (*sj_no89k*), all nodes of in_degree 10,000 or more depleted (*sj_no10kplus*), or all nodes of in_degree 1,000 or more removed (*sj_no1kplus*) from *sj*.

References

1. Traag, V.A., Waltman, L., van Eck, N.J.: From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports **9**(1) (2019). DOI 10.1038/s41598-019-41695-z