Refutations to "Refutations on Debunking the Myths of Influence Maximization: An In-Depth Benchmarking Study"

Akhil Arora, Sainyam Galhotra, Sayan Ranu

Recently, groups in University of British Columbia (headed by Lakshmanan et al.) and Nanyang Technical University (Xiao et al.) have published refutations on our benchmarking study as a technical report. In this report, we present our response to those refutations. Our comments are based on the technical report available at https://arxiv.org/pdf/1705.05144.pdf (version 3)

Before we start, let us give the following background information that sets the context.

1. CELF++ is a paper (more than 260 citations) authored by Amit Goyal, Wei Lu, and Laks V.S. Lakshmanan. CELF++ claims to be 35%-50% faster than CELF. In our study, we firmly establish that this claim is not true.
2. Our paper has been independently verified by SIGMOD reproducibility committee and has received the "SIGMOD Reproducible" tag.

Refutation 1: "Flawed" experimental design (Sec 1.2 and 2.2.1)

- The authors make this claim based on the following statements,
  "By construction, then different algorithms are not held to the same bar w.r.t. expected spread."

  "Thus, any comparison of the running times of different IM algorithms based on such an algorithm specific "near-optimal" spread necessarily holds the algorithms to different bars!"

  We asked the following question in our benchmarking study: Suppose we want to extract best possible quality from an IM technique. In such a case, how do these techniques scale against Seed Set size (k) (Figures 6 and 7 in our paper)? Our experimental design models this question and provides the answer.

  Certainly, Algorithm A being faster than Algorithm B does not mean A is better than B since, as the tech report points out, A may achieve a much lower spread. **Neither have we drawn any such conclusion in our paper.** We would like to point out that **a plot should not be confused with a conclusion. A plot presents data and a conclusion is drawn by analyzing this data.** For this precise reason, IMRank, which is significantly faster than IMM, have been dropped from the analysis on larger datasets (Table 3), because its spread is too low to be competitive. Neither does IMRANK (or any of the faster but spread-wise significantly inferior such as IRIE) feature in the decision tree (Figure 11 b).

  We only compare techniques that are comparable.
  - CELF and CELF++: In this comparison, we have pointed out an incorrect claim that has propagated through the IM community for more than 6 years! They are comparable in

terms of efficiency since CELF and CELF++ are guaranteed to provide the same expected spread under identical no. of MC simulations.

- TIM+ vs. IMM: We also compare TIM+ and IMM since they are based on similar frameworks. Furthermore, you will clearly see that TIM+ and IMM have almost identical spreads.
- SIMPATH Vs. LDAG: We compare the scalability of running times of these two techniques at the same parameters that were used in the SIMPATH paper and show that at higher values of k LDAG is faster. This is an observation that has not been brought out earlier and the statement made in the SIMPATH paper that *"Through extensive experimentation on four real data sets, we show that SIMPATH outperforms LDAG, in terms of running time, memory consumption and the quality of the seed sets."* (Conclusion in SIMPATH paper) may not always be true. Note, we discuss the correctness of the SIMPATH running times later in this draft.

In addition, we highlight several statements from our paper that clearly shows we are sensitive to the trade-off between time, spread and memory footprint.

- Sec: 5.3: "In LT, TIM++ is marginally faster than IMM while providing almost identical spreads."
- Sec 7: "When main memory is scarce, EaSyIM, CELF, CELF++ and IRIE provide alternative solutions. Among these, EaSyIM easily out-performs the other three techniques in memory footprint, while also generating reasonable quality and efficiency."

- The above argument justifies the experimental design for the research questions we ask in our study. Next, we would like to point out some contradictory views held by the authors of technical report. As a rectification of our "flawed" experiments, the technical report suggests that the running time of two techniques should be compared by fixing their spread to the same value, i.e., set them to the same bar. Here, we point out that the authors of this tech report themselves have not followed this procedure they are advocating for.
  - The SIMPATH paper (shares three common authors with the tech report) follows the exact same experimental setup as ours. The authors first choose the optimal parameters for each of the techniques they benchmark. In Fig. 3 of SIMPATH, the authors study the spread against number of seeds, and in Fig. 4, they study the growth of running time against number of seeds. Based on these plots, they draw their conclusion. This is identical to our setup (Figures 6 and 7 of our paper are identical in design to Fig. 3. and Fig 5. SIMPATH), which has been termed "flawed". To elaborate further, SIMPATH concludes it outperforms LDAG and CELF. The performance of LDAG can be controlled using a parameter θ. If we fix spread to a certain value X, it may very well be possible that there exists a θ, where LDAG achieves spread X at a lower running time than SIMPATH. Similarly, why did they not try smaller number of MC simulations for CELF? The same argument that they applied on our work applies to SIMPATH as well. Furthermore, how do you even select X? If X is set to be too low, then the random seed selection algorithm would be the best. If X is set too high, then many techniques may not even be able to attain that spread and therefore cannot be directly compared. If X is set somewhere in the middle which is attainable by all techniques, we may get incorrect results where IMRANK or IRIE would be termed best.

- In the comparison between TIM+ and IMM (shares one author with the tech report), they DO NOT first fix the spread and then find the time required to achieve that spread. Rather they compare the running times under a set of parameter values, wherein both techniques could achieve different spreads (Fig 7 and 9 in the IMM paper).
- To give one more example from the IM domain to substantiate that our experimental methodology is fairly standard and commonly practiced, we quote the following paper: "Sketch-based Influence Maximization and Computation: Scaling up with Guarantees, Cohen et al., CIKM 2014". As in our paper, SKIM compares its running time and spread quality with TIM+ at some chosen parameter values. Once again, they do not fix spread and then compare running times.

- To make our argument even more generic, consider any algorithm that has a quality vs. time tradeoff (Ex: a classification algorithm). When you compare the performance of two such algorithms, say A and B, you first select the optimal parameters for A and B based on some metric of your choice, compute their qualities at the optimal parameter values as well as their running times. Based on the results you get, you draw some conclusion. You don't fix the quality (such as f-score) to some particular value and then find out the training time taken to achieve that accuracy. Furthermore, as we have already pointed out, how do you even fix the quality bar?

  As an afterthought, to further assess the gravity of this refutation on experimental design, it challenges a majority of research work published in computer science. To this end, let us see how comparisons are performed in the literature in general. First, the parameters of the proposed algorithm (say A) are tuned to achieve its best performance. The best performance can either be superior quality/efficiency alone, or a best trade-off of both. While performing comparisons with the state-of-the-art, A would generally use the parameters recommended by the former. As A tuned its parameters to achieve its best performance, the parameters recommended by state-of-the-art are also (usually) a result of a similar exercise. We can see that some of the work published by the authors who have written this refutation follows a similar approach. SIMPATH tunes its own parameters, and then compares with LDAG using the recommended parameters by LDAG. Even TIM++/IMM, use the ϵ value of their choice, and compare against heuristics like IRIE and SIMPATH using their recommended parameters. In essence, there is no dearth of experimental designs similar to the one adopted by us in the published computer science literature. Moreover, there is no surprise in believing that even some of the research performed by the readers of this refutation would have also performed a similar exercise. This leads us to the question if *Lu et al. are trying to claim that the experimental design used by a majority of computer science research community (including their own) is wrong?*

Refutation 2: Irreproducible results (Sec 2.3)

- **Procedure:** Most of the algorithms have a parameter with which the quality of spread can be controlled. Let us call this variable X and assume that when X goes up, the spread goes up as well. Almost always, the running time goes up with X as well. Our goal was to first find the value of X* where the spread is highest. Then, reduce X such that the average quality obtained at X is close enough ("near optimal" is the term we use in our paper) to the average quality

obtained at X*. Since spread is computed based on MC simulations, each run with X* produces a different spread value, and prone to outliers. Therefore, we first compute the *standard error* of the mean spread at X*. It is computed by drawing repeated samples of some bin size b from a population, compute the mean for this sample set, and then finally the standard deviation of these means is the standard error. This procedure is done using the standard boot-strapping algorithm and the pseudocode is provided below. Once the standard error at X* is found, we apply the one-standard-error rule, wherein we choose the lowest X whose mean spread is within one standard error of the mean spread at X* (i.e., 1 standard deviation of the means of each sampled bin). In our experiment, we choose a bin size of 300. In the table below, we show the results for other bin sizes ranging from 100 to 400 for IMM. As evident, the parameter values are not much sensitive to the bin size.

- Pseudocode:

Algorithm 1 Calculate standard error

Input: List of MC simulation spread samples L,# bins, bin size b
Output: expected spread σ, std-err sd

```
i ← 0
avgList = []
while i ≤ # bins do
  j ← 0
  val ← 0
  while j < b do
    f ← random element from L
    val ← val + f
    j ← j + 1
  end while
  avgList.append(val/b)
  i ← i + 1
end while
σ ← Compute average of avgList
sd ← Compute standard deviation of avgList
```

Optimal ϵ value for IMM, Bin size on columns	100	200	300	400
IC	0.05	0.05	0.05	0.05
WC	0.1	0.1	0.05	0.05
LT	0.15	0.1	0.1	0.1

- **Why set parameters based on one-standard-error rule?** As mentioned, each run produces a different spread and the overall spread computation is a randomized procedure. Thus, it is important to know the confidence interval around the mean estimated spread. Standard error allows us to know the confidence interval and measures the accuracy with which a sample represents a population. We also note that the one-standard-error rule has been used in the literature, most notably in Classification and Regression Trees, and we provide few examples below.
  - Page 80 in Classification and Regression Trees by Breiman, Friedman, Stone & Olshen (1984)
  - Page 415 in Estimating the Number of Clusters in a Data Set via the Gap Statistic by Tibshirani, Walther & Hastie (JRSS B, 2001) (referencing Breiman et al.)
  - Pages 61 and 244 in Elements of Statistical Learning by Hastie, Tibshirani & Friedman (2009)

    - Page 13 in Statistical Learning with Sparsity by Hastie, Tibshirani & Wainwright (2015)
- We realize that our description of this procedure is not detailed enough in the paper and this is an implementation level detail that we have missed out on specifying in the Appendix along with an explanation of Fig. 12. The tech-report observes different standard deviations because they have computed standard deviation of the entire distribution. We welcome anyone to repeat our experiments and check if they are reproducible or not.
- Finally, we do not propose our parameter selection algorithm as a general-purpose algorithm for any dataset or any network. It is something that works well empirically on the datasets and models used for our benchmarking study.

Refutation 3: No definition of "reasonable time limit" (Sec 2.2.3)

- We point readers to footnote 4 where we cite two examples of unreasonable computation times. In M2 (Section 6), where we discuss the importance of number of MC simulations in CELF/CELF++, we give one more example stating that although 20k simulations is desired when the number of seeds is large, CELF takes 80 hours to finish even on the relatively small dataset of NetHept.
- Having said that, we find it troubling to note that the authors have a separate set of bars for our work and theirs. Specifically, we quote two statements from Sec. VI-C of the SIMPATH paper (3 common authors with the tech-report)
  "Due to MC-CELF's lack of efficiency and scalability, its results are only reported for NetHEPT and Last.fm, the two datasets on which it can finish in a reasonable amount of time."

  "Note that the plots for NetHEPT and Last.fm have a logarithmic scale on the y-axis. MC-CELF takes 9 hours to finish on NetHEPT and 7 days on Last.fm while it fails to complete in a reasonable amount of time on Flixster and DBLP."

  In neither case, the SIMPATH paper defines what "reasonable" time limit means.

Refutation 4: Computing near-optimal spread (Sec 2.2.2)

- First, this analogy does not apply to our problem since we do not compute the standard deviation of the entire distribution. Rather, we use standard deviation to compute the standard error of the optimal mean spread.
- For the sake of argument, let us assume that we do compute standard deviation on the entire distribution. The graph portrayed in this example (Fig 3. of tech report) is not a power law graph and it is well known that online social networks follow a power law. In our humble opinion, this pathological case is of only theoretical interest without any practical importance.
- Let us now point out the factual issues as we observed them in our experiments on real data. Outliers with extreme values do occur as a result of the MC simulations. It is thus important to understand the error associated with the mean spread. Our parameter selection design models these requirements.

Refutation 5: TIM+ vs IMM MISCLAIM 1 (Sec 3.1)

- In this claim, we make the statement that TIM+ and IMM fails to scale beyond HepPh in terms of memory consumption at $\epsilon = 0.05$ in the IC model. The tech-report finds a problem with this statement since we don't allow enough error in the quality of TIM+ or IMM. They argue that we

should instead increase error (controlled by ϵ) in the spread quality and let it scale better. By this logic, we can even make CELF and CELF++ scale by setting the number of MC simulations to 100 and therefore allow a high error.

- **Why is it important to set a high quality bar of $\epsilon = 0.05$?** Our goal is to identify the technique that provides a high spread and scales well. Towards that end, notice the performance of PMC (N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. AAAI 2014.), which produces a spread quality as good as IMM and TIM+, runs 100 times faster than IMM and TIM+, and consumes up to 100 times less memory. More simply, it produces better performance that IMM and TIM+ in IC. Unless we set ϵ to a low value and ensure that TIM+ and IMM are allowed to produce high spreads, there is no way for us to draw this conclusion that PMC is as good or better in all aspects.
- This claim creates an impression that the techniques, IMM and TIM+ were forced to produce highly accurate results. We would like to state that the parameter was chosen according to the one-standard-error rule. If the technique provides high quality seeds with higher epsilon, we do consider that particular value. Eg. In case of LT model, the TIM+ algorithm gave good seeds at an epsilon value of 0.35. On the other hand, for IC model the techniques (TIM+ and IMM) generate good seeds only at lower values of epsilon. Overall, the parameter estimation procedure does not discriminate against any specific technique.

Refutation 6: TIM+ vs IMM MISCLAIM 2 (Sec 3.1)

- First, let us correct **a factual error** in the technical report by Lu et al.

  "Arora et al. compare the empirical running time of TIM+ and IMM under the LT model by setting ϵ = 0.1 for TIM+ and ϵ = 0.05 for IMM, and they conclude that TIM+ is empirically more efficient than IMM under the LT model."

  In reality, we use $\epsilon = 0.1$ for IMM and $\epsilon = 0.35$ for TIM+ as clearly documented in Table 2 of our paper. Thus, the above statement misrepresents facts. This incorrect statement of the paper is repeated once more where the technical report says

  "In contrast, Arora et al. compare the efficiency of TIM+ and IMM under the LT model with ϵ = 0.1 for TIM+ and ϵ = 0.05 for IMM"

  While counter-opinions and deeper discussions on a published work are always welcome, they must be based on a thorough and careful reading of the paper.

- In our study, we show that IMM is not strictly faster than TIM+. More specifically, in LT model, TIM+ is marginally faster. The tech-report says that this claim is invalid. They come to this conclusion since we evaluate TIM+ and IMM on different ϵ values, where ϵ provides the worst-case theoretical guarantee on the spread quality. They reason, that TIM+ and IMM must be compared at the same ϵ values. We now point out several loopholes in this argument.
  - In Sec 2.2.1, the tech report argues that all techniques must be compared by fixing their spreads to the same value. If we are indeed to follow that procedure, then we must allow comparing TIM+ and IMM at different ϵ values. However, for the specific TIM+ vs. IMM comparison, the tech-report suggests comparing them at same ϵ values, and any other form is simply invalid according to them. This is a clear case of presenting contradictory arguments.

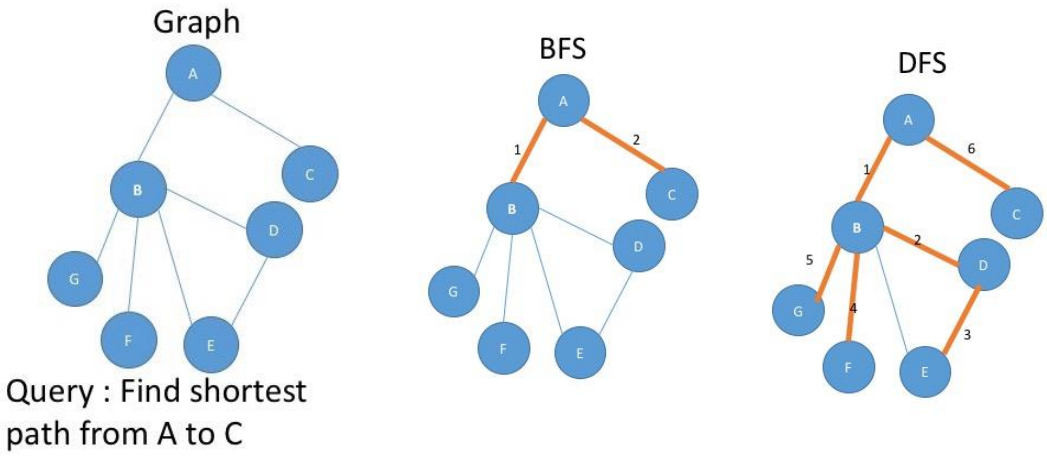
- The tech-report suggests comparing TIM+ and IMM at same ϵ value since that ensures the same worst-case guarantee for both. An empirical benchmarking study does not evaluate theoretical guarantees. Rather, it compares the actual empirical results obtained by each technique.
- One major application of IM is on viral marketing. Consider a person from the advertising domain who wants to use an IM technique as a black box and better advertise his/her product. Such a person is much less interested in the worst case theoretical guarantee of the technique when compared to the actual spread obtained. We compute these spreads by benchmarking on a wide array of datasets and diffusion models and make the decision process easier.

- The tech-report further argues that if at all we should evaluate their empirical accuracies, we should ensure that both TIM+ and IMM have the same sample set sizes.
  - Indeed, if sample sets are identical, both TIM+ and IMM would produce same results. **However, neither TIM+ nor IMM expose any control over the selection of sample sets**. The only way to control sample sets is to modify the code of TIM+ and IMM, and as we have stated in our experimental setup, this is not the goal of our study. Our goal is to study performance by only varying parameters that are exposed by the techniques being benchmarked (Sec 5.1, 3rd para). This is in line with standard practice in our community, where we observe the performance of a software through the controls exposed by its API. We do not modify the code.
  - The only indirect control over the size of sample sets that is exposed by TIM+ and IMM is through ϵ. Our study identifies how ϵ affects the performance in terms of quality, running time and memory consumption. The key highlight of our study is that while at identical ϵ, IMM is faster than TIM+, the quality of TIM+ is better. To achieve the same level of quality, TIM+ at times is faster than IMM and this is a result that has not been highlighted in the literature.
  - Finally, if it is obvious that at same ϵ TIM+ would produce better spreads (as indicated in the tech-report), it should also be obvious that at same ϵ IMM would be faster as the number of sample sets is smaller in IMM. However, we see the authors of the tech-report (common with TIM+ and IMM) contradict themselves since they have conducted this experiment of ϵ vs running time (Fig. 6a, 6b in IMM paper).
- Finally, the tech-report draws an analogy of Chebyshev Vs. Chernoff bounds. We have the following points to make on this analogy.
  - **Importance of empirical analysis:** This is an invalid analogy since we benchmark TIM+ and IMM from an empirical standpoint as two software tools, whereas Chebyshev and Chernoff provide worst-case theoretical bounds on the number of samples required to generate a desired level (ϵ) of quality. Empirical evaluations of memory consumption, computation time, etc. might not correspond to the worst-case values posed by these theoretical bounds. Rather, they may be affected by the empirical values of the number of samples required for a given ϵ value. To substantiate this argument, we provide a simple example.

    BFS vs DFS: Suppose we have two choices, BFS and DFS to answer reachability queries. Although the worst-case complexities of the two techniques are same, it is possible that empirically BFS is better than DFS for the kind of queries our systems asks and vice versa. E.g. Consider the graph in the figure below and query to find shortest path from A to C. If we employ BFS, we will end up at the solution in maximum of two edge expansions. On the other

hand, it is possible that DFS makes 6 edge expansions to reach the solution (It is possible that DFS runs faster than BFS in a particular run but even in expected sense, BFS will perform better for this query.). Clearly, the BFS algorithm "might" perform better when the path length for the queries is small. While DFS might be better in long distance queries.

It is clear from this example that worst-case theoretical guarantee alone may not provide the full picture and it is important to conduct experiments to observe the empirical behavior of two competing techniques. Our study achieves this goal and provides additional insights by comparing various techniques empirically keeping aside their "worst case" asymptotic complexities. With respect to TIM+ and IMM, what epsilon should we choose so that high spreads are observed across a range of datasets? How do the running times compare when their qualities are similar? How much memory do they consume? These are all valid questions and our study answers them.



- It is well known that Chebyshev is weaker than Chernoff when applied on same random variable. This means to get similar accuracy (analogous to spread), we get different epsilon values for the two techniques. This contradicts the argument of using same epsilon to compare the two techniques. Specifically, earlier the technical report argued for comparing techniques by setting their spreads to the same value. On the other hand, for TIM+ and IMM, the report argues for comparing them at same ϵ and therefore different spreads. This is a clear contradiction of their earlier statement.
- The tech report states an example where the task is to consider 10K samples and Chernoff based approach chose ~18K and Chebyshev based approach chose ~11K samples. We completely agree that Chernoff is better as it considers more samples but if these excessive samples do not help in providing (substantially) better quality seeds, then it is useless. Now Chebyshev is efficient in this scenario because the task is to select a technique which gives more than 10K samples as quickly as possible. If a particular technique uses Chernoff and needs to generate 18K samples even when there is no gain in quality with this additional work, it cannot be considered superior over an approach that provides the right number of samples.

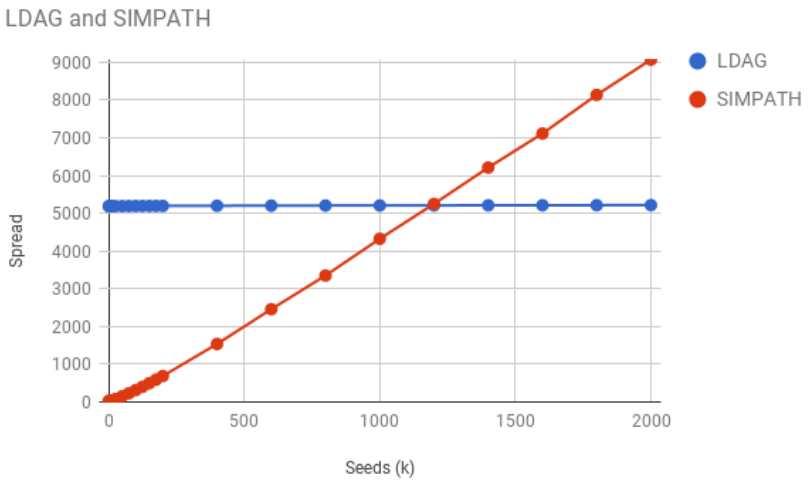
Refutation 7: SIMPATH Vs. LDAG (Sec 3.2)

- **Acknowledgement:** The running times reported in our work for SIMPATH in the DBLP and YouTube datasets are incorrect. The correct running times are provided below. In our SIGMOD presentation, we started our talk by stating this fact (Xiaokui Xiao is a witness to this). This information has also been posted on our websites.
- **Source:** Let us now point out the source of this error. We used the SIMPATH code released by the authors. The code makes an assumption that node IDs are positive numbers. **This assumption is not mentioned anywhere in the README specifications released by the authors.** The only way, this assumption can be identified is by studying the SIMPATH code containing more than 1000 lines. To make matters worse, some datasets such as NetHep and HepPh ran perfectly fine with node IDs starting from 0. Clearly, this is a classic case of an undocumented assumption in the SIMPATH code, which was identified due to our study.
- **Sharing datasets with SIMPATH authors:** We had sent an email to the authors of SIMPATH on April 2, 2016 (email present here) when we could not reproduce few results on a dataset whose first line had node id 0 and the authors still didn't point us that it could be a source of error. It forces us to infer that they themselves were unaware of such implicit assumptions they made in their code. It was introspection after our work that they were able to understand their code better.
- **Obfuscation of facts:** In the tech-report, the authors do not mention anywhere that this assumption is undocumented in their README specifications. Neither do they mention we reached out to them with a dataset containing node IDs starting from 0. They simply state, we supplied "incompatible datasets", although we supplied datasets that are perfectly compatible with the specifications released by them. While we fully understand that this is research code and human errors are always possible, not stating the above facts, in our humble opinion, is obfuscation of facts.
- **Impact:** Let us now discuss the impact of the correct running times on the SIMPATH Vs LDAG comparison. The corrected Table 4 from our paper is provided below. As can be seen, SIMPATH is slower than LDAG across all five experiments. Furthermore, there is no change on Figures 10 a and b, since SIMPATH ran fine one these two datasets. Consequently, **the conclusion drawn in our paper with respect to this comparison remain absolutely unchanged.**

Algorithm	NetHept	NetHept-P	HepPh	DBLP	DBLP (large)-P
LDAG	0.37 min	0.32 min	1.5 min	5.5 min	26.4 min
SIMPATH	1.5 min	1.1 min	8.1 min	6.4 min	34.6 min

- **MISCLAIM 4 in tech-report:** MISCLAIM 4 in the tech-report is incorrect since SIMPATH remains slower than LDAG on DBLP even after correcting the code.
- **MISCLAIM 5:** It is unclear why this is being reported as a mis-claim. The authors simply provide a justification of why they chose not to vary the number of seeds beyond 50 in their paper. That does not invalidate our claim. Our paper points out that at 150 seeds and beyond SIMPATH is slower than LDAG on DBLP and NetHept datasets. This explicit point has indeed not been pointed out in the literature.
- **MISCLAIM 6 and Performance in YouTube:** Scalability does not imply better performance (say efficiency, memory-consumption etc.) for a particular choice of parameters. However, it is aimed at measuring the variation in the change in performance when the parameters are varied. To give

a concrete instance, the technical report says since SIMPATH is faster at K=200 for YouTube, it refutes our claim that LDAG is more scalable and robust. To point out the error in this logic, we present the scalability of LDAG and SIMPATH on YouTube below. It is evident that LDAG becomes comparable in running time to SIMPATH at 1200 seeds and better beyond that, and thus is rightly deemed to be more scalable in our paper. Note that the cross-over points in terms of the number of seeds (k) is not unreasonable with respect to the number of nodes in the YouTube network. From the results portrayed in our paper and in Fig. 5(a) and Fig. 5(c) by Lu et al., it is clear that LDAG scales better when compared to SIMPATH.



Refutation 8: Memory efficiency of EasyIM (Sec 4.1)

- The tech-report claims EasyIM is not the best algorithm for large datasets since it did not finish on Orkut. This is true and we do not recommend EasyIM as the most scalable technique with respect to computation time. Rather, our recommendation is on scenarios where the bottleneck is main memory space (such as a commodity laptop). It can be inferred from Fig. 8 in our paper that EasyIM has 100 times smaller memory footprint than IMM and 1000 times smaller than TIM+.
- This mis-claim is largely a critique of the decision tree diagram provided in our paper (Fig. 11b). We would like to mention that the decision tree is by no means an Oracle that always outputs the best IM technique given the dataset, model and constraints involved. The idea behind this decision tree is to provide some guidelines and it should be treated as such. It should not be confused as a theorem.

Refutation 9: Mis-claim 10 (Sec 4.2)

- The tech-report argues that our statement that Celf of Celf++ is gold standard is flawed. Indeed, this statement is flawed, and that is why we state this as a myth. Thus, we don't see a difference of opinion here. However, we are intrigued at why this is classified as a mis-claim.

- We feel that it is important to highlight this common misinterpretation (or myth) and bring it to the attention of IM community since several papers use CELF/CELF++ for benchmarking (listed below).
  - S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In CIKM, pages 509–518, 2013.
  - A. Goyal, W. Lu, and L. V. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In ICDM, pages 211–220, 2011.
  - K. Jung, W. Heo, and W. Chen. IRIE: Scalable and robust influence maximization in social networks. In ICDM, pages 918–923, 2012.
  - J. Kim, S.-K. Kim, and H. Yu. Scalable and parallelizable processing of influence maximization for large-scale social networks. In ICDE, pages 266–277, 2013.
  - A. Khan, B. Zehnder, and D. Kossmann. Revenue maximization by viral marketing: A social network host's perspective. In ICDE, pages 37--48, 2016.
  - Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, and J. X. Yu. Influence maximization over large-scale social networks: A bounded linear approach. In CIKM, pages 171–180, 2014.
  - H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In SIGMOD, pages 695–710, 2016.
  - Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In SIGMOD, pages 75–86, 2014.

Refutation 10: MISCLAIM 11 (Sec 4.2)

- In this mis-claim, the tech-report alleges that our paper "grossly oversimplifies reality" by assigning a uniform probability of 0.1 to all edges and calls it the "general IC model". This allegation is not true. To prove that this mis-claim is incorrect, we simply quote statements from our paper.
  - Following excerpt is from Sec 2.1.1

2.1.1 Independent Cascade (IC)

• **Constant:** In this model, each edge $W(u, v)$ has a constant probability p. In vast majority of the IM techniques, p takes the value of either 0.01 or 0.1 [4, 9–11, 14, 17]. Additionally, some techniques use a spectrum of values for $p \in [0.01, 0.1]$ [5, 24].

• **Weighted Cascade (WC):** In WC, $p(u, v) = W(u, v) = \frac{1}{|\mathbf{In}(v)|}$. In other words, all incoming neighbors of v influence v with equal probability. As a consequence, it is easier to influence low-degree nodes than high-degree nodes. [4, 5, 7–11, 14, 16, 17, 26, 27]

• **Tri-valency Model:** In this model, the probability (or weight) on an edge is chosen randomly from a set of probabilities. For example, the weight may be chosen randomly from the set $\{0.001, 0.01, 0.1\}$. [4, 7, 16]

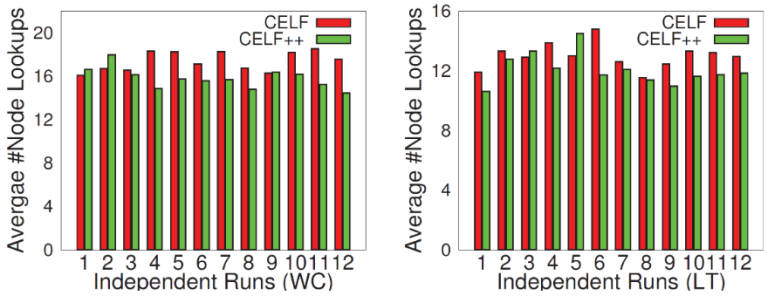
  - In Section 5.1, we clearly mention that we are using the IC-constant model and it is denoted as IC. By no stretch of imagination, this can line can be inferred as using the generic IC model.

Information-Diffusion Models: We incorporate the use of three diffusion models, namely – IC-constant denoted using IC, IC-WC denoted as WC and LT-uniform denoted as LT (Sec. 2.1). The IC model is used with a constant probability $p_{(u,v)} = 0.1$ assigned to all the edges of the network. As opposed to the IC and WC models, the LT model requires an additional parameter. Apart

- Furthermore, to leave no room for ambiguity, we clearly mention in Sec 2.1 that ideally these edge-weights should be learned. However, due to lack of training data, such an exercise is often not possible.

Refutation 11: CELF Vs. CELF++ (Appendix B)

- We welcome the authors of CELF++ (and the tech-report) acknowledging that their CELF++ paper is based on an incorrect conclusion. However, we are not fully convinced on the explanation provided.
  - The tech-report states that their experiments ran into "noise" and this noise may stem from various issues including caching and core utilization. We present Fig. 13 from our paper below, where we plot the number of node look-ups performed by CELF and CELF++.



(a) Nethept (WC) (b) Nethept (LT)

Figure 13: **Comparing average number of node-lookups for CELF with CELF++ on the Nethept dataset over 12 independent executions.**

  - The number of look ups are independent of the hardware infrastructure or the state of the CPU at that time and directly affects the running time. Note that, CELF++ is not always better in terms of node-lookups when compared to the CELF algorithm. In fact, since in CELF++ you compute the spread twice for each node, even when the node-lookups are same, we believe the run-time of CELF would be faster/better when compared to CELF++. The explanation in the tech report fails to explain this behavior.
  - More importantly, we quote the following line from Section 3 of the CELF++ paper.

    "*This is due to the fact that the average number of "spread computations" per iteration is significantly lower.****"***

    The number of spread computations is a function of the number of node look-ups and independent of hardware resources. We are therefore unable to find any basis based on which the above claim could have been made.

- Finally, we would like to provide some background on the CELF Vs CELF++ comparisons. We pointed out these discrepancies based on node look-ups to Laks et al. on May 17, 2016 much before we even submitted our paper to SIGMOD for review (link to email). The authors chose to not respond to our query till our paper got published roughly 8 months later. At that time, they justified not responding to us with the following statement (link to email).
  "*Being a researcher yourself, you may understand that we receive several queries about our papers, and its not possible to respond to all of them...*"
- **Importance of a neutral point of view:** Finally, we would like to highlight certain aspects of the technical report that may indicate its neutrality. The technical report uses various negative words to describe our work. Our experimental design has been termed "incorrect" and "flawed" at multiple places. Furthermore, words like "profound", "gravity", "seriousness", etc. have been used frequently while discussing our results. In contrast, notice the language used while acknowledging their own errors in CELF++. Our paper comprehensively establishes CELF++ is at par with CELF. **This invalidates the entire basis of their own publication (3 common authors with this technical report). Yet, the authors simply state their results on CELF++ *"ran into noise".*** Kindly note that their paper possesses approximately 260 citations based on an incorrect claim and appears in the prestigious WWW conference. **With this context, we leave it to the readers to draw their own conclusions on the neutrality of the technical report.**

- **Appendix:**
  - The authors of this technical report had earlier sent an email to the SIGMOD PC chair stating that our paper possess serious flaws. The refutations detailed in the technical report available at https://arxiv.org/pdf/1705.05144.pdf (version 3), are more or less based on the set of 11 flaws that they had pointed out in their email. We had then emailed our response to the SIGMOD committee and our paper was discussed again by the SIGMOD PC Chairs, Vice-chairs, and the original reviewers of our paper, and was still marked to be fit and worthy for a place in the proceedings of SIGMOD 2017. The links to the emails are provided below.
    - Email by Lakshmanan et al.
    - Email by us to Prof. Suciu (SIGMOD PC Chair)
    - Response by Prof. Suciu after reading our rebuttal

    (Permissions were taken from Prof. Dan Suciu before releasing the above emails publicly.)

  - It has also been found that the running time of LDAG on the DBLP dataset reported in the SIMPATH paper (authored by Lu et al.) is incorrect. The authors have accepted this error and suspect hardware noise to be the reason behind this incorrect running time. Interestingly, both in CELF++ and SIMPATH, the "noise" have falsely created a more positive image of the authors' techniques than reality.