Perturb and Combine to Identify Influential Spreaders in Real-World Networks

Antoine Jean-Pierre Tixier, Maria Evgenia G. Rossi, Fragkiskos Malliaros, Jesse Read, Michalis Vazirgiannis

Perturb and Combine to Identify Influential Spreaders in Real-World Networks

Antoine J.-P. Tixier[1], **Maria E.G. Rossi**[1], **Fragkiskos D. Malliaros**[2], **Jesse Read**[1], **Michalis Vazirgiannis**[1]

[1]Computer Science Laboratory, École Polytechnique, France
[2]Center for Visual Computing, CentraleSupélec, University of Paris-Saclay and Inria Saclay, France

Abstract

Graph degeneracy algorithms were recently shown to be very effective at detecting the influential spreaders in a network. However, degeneracy-based decompositions of a graph are unstable to small perturbations of the network structure. At the same time, it is well-known in Machine Learning that the performance of unstable algorithms can be greatly improved by using Perturb and Combine (P&C) strategies. Motivated by these observations, we propose a P&C procedure for networks that (1) creates many perturbed versions of a given graph, (2) applies the same algorithm separately to each graph, and (3) combines the results. Experiments conducted on benchmark datasets of real-world networks reveal that our strategy improves the performance of all algorithms tested. Moreover, this performance boost can be obtained at almost no extra cost through parallelization. We finally provide insights as to why our strategy is effective from a theoretical perspective. To the best of our knowledge, this work is the first application of P&C to networks.

Introduction

Graphs (or networks) are widely used to represent real-world data. Influential spreaders can be defined as the nodes that are able to diffuse information to the largest part of the network in a given number of time steps. Influential spreader identification finds applications in a variety of fields, from epidemiology (Hoppensteadt 1975) and marketing (Leskovec, Adamic, and Huberman 2007) to NLP (Tixier, Malliaros, and Vazirgiannis 2016).

Graph degeneracy algorithms are useful for all tasks involving the identification of the densest areas of a graph. They also have well-understood properties, can be implemented efficiently, and scale to large networks. As a result, they have been applied to many tasks, ranging from graph compression and clustering to finding correlated genes and detecting fraudsters. In particular, they have recently been shown very effective at locating the good spreaders in a network (Kitsak et al. 2010; Malliaros, Rossi, and Vazirgiannis 2016).

The unweighted and generalized k-core algorithms (Seidman 1983; Batagelj and Zaveršnik 2002) are the oldest and most famous members of the graph degeneracy family. Their high success has motivated many extensions, such as the k-truss (Cohen 2008), nucleus (Sariyuce et al. 2015), and k-peak (Govindan et al. 2017) decompositions.

The k-core algorithm assigns to each node of a graph $G(V, E)$ a non-unique score, the core number, corresponding to the highest order of a core the node belongs to. A core of order k of G is the maximal subgraph of G in which every vertex has at least degree k. The generalized k-core algorithm works for any local monotone vertex property function. It runs in $\mathcal{O}(|E| \log |V|)$ while the basic version is linear in the number of edges. When used with the weighted degree as the vertex property function, we simply refer to the generalized k-core algorithm as weighted k-core. Nodes with high core numbers have the desirable property to not only have many connections, but also to form tightly interconnected subgraphs together with their connections. (Kitsak et al. 2010) showed that core numbers are more strongly correlated with spreading influence than degree or PageRank scores.

Motivation. Two key observations provided the motivation for this study: **(1)** graph degeneracy algorithms are highly unstable to small network perturbations, i.e., removing a tiny fraction of edges or nodes from the graph can significantly change node scores (Adiga and Vullikanti 2013; Goltsev, Dorogovtsev, and Mendes 2006). **(2)** In Machine Learning, it is well-known that unstable algorithms, i.e., algorithms for which small changes in the training set result in large changes in predictions, greatly benefit from Perturb and Combine (P&C) strategies (Breiman 1996b).

Bootstrap **agg**regat**ing** or bagging (Breiman 1996a) is one of the most popular of the P&Cs strategies. It is actually one of the two key ingredients of the acclaimed Random Forest model (Breiman 2001). Bagging simply consists in training a model in parallel on bootstrap samples of the training set. Each bootstrap sample is a *perturbed* version of the original training set which is generated by drawing from it with replacement until a set of the same size is obtained. To issue a forecast, the predictions of all the models are *combined*, e.g., through averaging in regression and majority voting in classification.

Goal of this paper. Motivated by the two aforelisted observations, we posit that like unstable algorithms in Machine Learning, ***degeneracy-based node scoring functions, and more generally, any unstable node scoring function, may benefit from using a Perturb and Combine (P&C) strategy***.

Main contributions. We propose what is, to the best of our knowledge, the first P&C strategy for networks. Our procedure features three simple steps: (1) create many perturbed versions of a given graph, (2) separately apply a node scoring function to each perturbed graph, and (3) combine the scores. By conducting experiments on several benchmark datasets of real-world networks, we show that the P&C scores allow to identify much better spreaders than the original scores, for all node scoring functions tested. Furthermore, our procedure is trivially parallelizable, so the P&C scores can be obtained at little extra cost. Finally, we define the bias and variance of a vertex scoring function, and explain from a theoretical perspective why P&C for networks is effective.

Perturb and Combine for Networks

Before going into the details of our strategy, we give the intuition for it through a simple example.

Motivational example

Look at the square and rectangle nodes in Figure 1. In the original graph, the square node is a member of the main core ($k = 4$), but a quick visual inspection reveals that this node does not lie in the most central part of the network and is not strongly attached to the main core. With a degree of only 4, the square node is actually one of the weakest members of the main core, i.e., removing only one of its connections would suffice in decreasing its score.

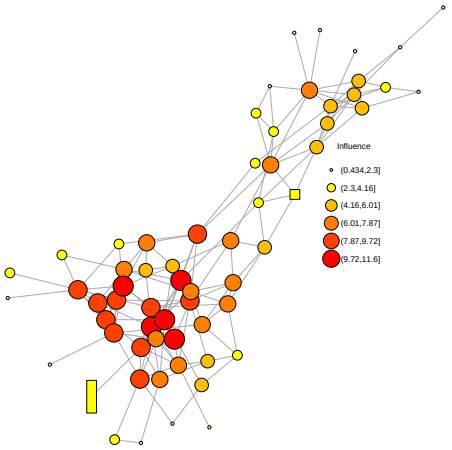


Figure 1: Spreading influence.The table below shows the original and P&C scores of the square and rectangle nodes, for k-core. The P&C score of a node is the mean of its scores in each of the 8 perturbed graphs. 'inf.' denotes true influence. Scores and influence are in different units, only ratios matter.

	or.	perturbed								agg.	inf.
square	**4**	3	3	3	2	1	2	3	2	**2.38**	4.09
rectangle	**1**	1	2	3	1	2	1	1	2	**1.62**	2.76

On the other hand, the rectangle node is part of the 1-core in the original network. This low score does not reflect the fact that this node has direct access to the very core of the network through its single connection. Should an epidemic be triggered from that node, it would probably be more severe than its low score suggests. To sum up, based on the original scores, the square node is 4 times more influential than the rectangle node. This is far from reality, as the ratio of the true influence scores is only $4.09/2.76 = 1.48$.

Looking at the scores obtained by the square and rectangle nodes in 8 slightly perturbed versions of the original network (a few edges added/deleted at random), we can observe that the rectangle node gets higher scores in most perturbed graphs, whereas the square node gets lower scores most of the time. Using the average of these 8 scores instead of the original scores is much closer to the true ratio: $2.38/1.62 = 1.47$.

Through this simple example, we can get a sense of how P&C can be beneficial. In short, perturbing the original network allows us to generate different configurations of it, which is akin to drawing from some true (but unavailable) underlying graph of which the original network is a single snapshot. Scoring each perturbed graph separately and combining back the scores can therefore be seen as *estimating the true scores of the nodes based on more evidence*. These ideas will be further elaborated in the Theoretical Analysis Section. We now provide details for each step of our P&C process.

Perturb

High-level framework. We used a standard framework for edge-based perturbation (Adiga and Vullikanti 2013). Adding/removing edges only (and not nodes) is preferable in our case as combining the results is more straightforward if all nodes exist in each perturbed graph. Let $G(V, E)$ be the original graph and $\mathbb{G}$ be a random graph model. The corresponding perturbation model $\Theta(G, \mathbb{G}, \varepsilon_a, \varepsilon_d)$ is defined as:

$$\mathbb{P}_\Theta\big[(u,v)\big] = \begin{cases} \varepsilon_a \mathbb{P}_\mathbb{G}\big[(u,v)\big], & \text{if } (u,v) \notin E \\ \varepsilon_d \mathbb{P}_\mathbb{G}\big[(u,v)\big], & \text{if } (u,v) \in E \end{cases} \quad (1)$$

where $\mathbb{P}_\Theta\big[(u,v)\big]$ is the probability of adding/deleting edge (u,v), $\mathbb{P}_\mathbb{G}\big[(u,v)\big]$ is the probability of selecting edge (u,v) according to the random graph model $\mathbb{G}$, and ε_a, ε_d are the probabilities of edge addition and deletion, respectively. By XOR-ing the original graph G with one realization $\theta \sim \Theta(G, \mathbb{G}, \varepsilon_a, \varepsilon_d)$ of the perturbation model, we obtain the perturbed graph $\tilde{G} = G \oplus \theta$. Depending on the random graph model $\mathbb{G}$ used, we obtain a different perturbation scheme.

Edge weight awareness. In the perturbation framework described above, edge weights are ignored, and edges can only be completely removed or created from scratch. To make our approach more flexible, and generalizable to weighted graphs, we created a variant in which an edge (u,v) can be considered for addition even if it already exists, and can remain in the graph even if it was selected for deletion. In such cases, we simply increment (respectively, decrement) the weight of (u,v) by one standard deviation of all edge weights. Since edges can be selected multiple times, any edge whose weight becomes negative is removed from the graph. In the rest of this paper, the δ_w Boolean parameter will indicate which of the two variants above is being used ($\delta_w = 1$ corresponds to the edge-weight aware scenario).

Note that in both scenarios, every time a new edge is added to the network, we sample its weight at random from the weights of the edges incident on its endpoints.

Random graph models. Plugging the Erdős-Rényi (**ER**) random graph model (Erdös and Rényi 1960) into our framework returns the *uniform perturbation model*. A node is randomly drawn with replacement from V with probability $1/n$. On the other hand, using the Chung-Lu (**CL**) random graph model (Chung and Lu 2002) gives the *degree assortative perturbation model*. A node is randomly drawn with replacement from V with probability proportional to its weighted degree. In that case, edges are more likely to be created/incremented and deleted/decremented between hubs. For both $\mathbb{G} = ER$ and $\mathbb{G} = CL$, self-edges are disregarded. That is, if we select two nodes u and v such that $u = v$, we discard the pair and select two other nodes.

Score

Each perturbed version of the original network is separately scored. Any node scoring function can be used, but as previously mentioned, P&C strategies work best in Machine Learning for unstable algorithms, so we expect our method to be best suited to *unstable* node scoring functions.

Combine

Following common practice (e.g., bagging regression trees), we compute the P&C score of a node as the *average* of its scores in each perturbed graph.

Parameters and parallelization

Our P&C strategy has the following 5 parameters: the number M of perturbed versions of the original graph to generate, the random graph model $\mathbb{G}$, the edge addition and deletion probabilities ε_a and ε_d, and edge weight awareness (δ_w).

Clearly, the *Score* step of our procedure is trivially parallelizable. The P&C scores thus do not take more time to obtain than the original scores, provided that M workers are available. The only additional cost comes from the initial *Perturb* step, but it can be implemented efficiently.

Experiments

Datasets

Social networks. We used a set of 3 well-known, publicly available large social networks (Leskovec and Krevl 2014) which we briefly describe in what follows. Descriptive statistics can also be found in Table 1. Since these networks are unweighted, the weights of the edges were assigned as the maximum degree of their endpoints.

EMAIL-ENRON is an email communication network built from 500K messages. Each node represents an email address and there is an edge between two nodes if the users exchanged at least one email. EPINIONS is a who-trust-whom online social network constructed from the general consumer review site `epinions.com`. Edges represent trust relationships between users. WIKIVOTE was created by extracting all Wikipedia administrator elections and vote history data up to January 2008. It contains data from 2,794 elections with 103,663 total votes and 7,066 users.

Table 1: Statistics of the real-world social networks used in this study. diam., cu_m, cw_m and pr_m denote the diameter of G, and maximum k-core, weighted k-core, and PageRank scores of its nodes. τ is the epidemic threshold of G.

$G(V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	diam.	cu_m	cw_m	pr_m	$\tau \times 10^2$
EMAIL-ENRON	34K	181K	11	43	19K	15	8.4
EPINIONS	76K	406K	14	67	37K	3	5.5
WIKI-VOTE	7K	101K	7	53	16K	4	7.2

Word co-occurrence networks. We used the standard Hulth2003 dataset (Hulth 2003), which contains abstracts from the Inspec research article database. More precisely, we considered the 500 documents in the validation set and used the uncontrolled keywords assigned by human annotators as ground truth. The mean document size is 120 words, and on average, 21 keywords (unigrams) are available for each document. As a pre-processing step, we applied part-of-speech tagging and retained only nouns and adjectives, following common practice (Mihalcea and Tarau 2004). Finally, we stemmed words with Porter's stemmer. We then represented each document as a word co-occurrence network, using a sliding window of size 5. In such a network, two nodes are linked by an edge if the terms they represent co-occur within the window, and edge weights indicate co-occurrence counts. The average number of nodes, edges, and diameter of the networks were respectively 32, 155, and 3.6. Human keywords were also stemmed, but not filtered based on their part-of-speech tags.

Setup

We experimented with the k-core and weighted k-core node scoring functions. For the sake of comparison, we also included PageRank (Page et al. 1999), since it is robust to link-based perturbations (Ipsen and Wills 2006; Ng, Zheng, and Jordan 2001). Note that we used the weighted version of PageRank. In the remainder of this paper, cu, cw, and pr will denote k-core, weighted k-core, and PageRank.

For each function, we compared the spreading influence of the nodes with highest scores in the original networks to that of the nodes with highest P&C scores.

Social networks. Here, we define the spreading influence of a node v as the number of nodes infected at the end of a SIR epidemic triggered from v. The SIR epidemic model (Kermack and McKendrick 1932) is widely-used in the domain of influential spreader detection. As shown in Figure 2, it is a discrete time model which assumes that at every step, each node falls into one of the following mutually exclusive categories: Susceptible (S), Infected (I), and Recovered (R).

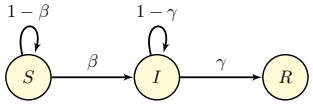


Figure 2: SIR epidemic model state diagram.

In the I state, a node is capable of contaminating its neighbors in the S state with probability β (infection rate). Then, at the next time step, it transitions to the R state with probability γ (recovery rate) or stays in the I state. Nodes in the R state are not able to get infected or transmit the disease anymore.

Initially, all the nodes are in the S state, except v which is in the I state. The process iterates until no new node gets infected for two consecutive steps. Following common practice (Kitsak et al. 2010; Malliaros, Rossi, and Vazirgiannis 2016), we average results over 100 epidemics started from v to account for the stochasticity of the process. We also set the infection rate β close to the epidemic threshold of the network $\tau = \frac{1}{\lambda_1}$, where λ_1 is the largest eigenvalue of the adjacency matrix (Chakrabarti et al. 2008); and the recovery rate γ to 0.8, as in Kitsak et al. (Kitsak et al. 2010). For cu and cw, we trigger an epidemic from each node in the maximal k-core subgraph. Since we use averaging as our combination strategy, the aggregated scores may be real numbers, so we round them to the upper integer before selecting the top core. For pr, we start an epidemic from the 100 nodes with highest scores. In each case, we average results over all trigger nodes to get a final performance score.

For exploration purposes, we experimented with the following P&C parameter values: $\varepsilon_a : \{0, 0.05, 0.1, 0.2\}$, $\varepsilon_d : \{0, 0.05, 0.1, 0.2\}$, $M : \{16, 64\}$, $\mathbb{G} : \{ER, CL\}$, and $\delta_w : \{0, 1\}$. Excluding the cases where $\varepsilon_a = \varepsilon_d = 0$, this made for 120 combinations.

Word co-occurrence networks. Here, we assume that the keywords of a document are the *influential* nodes of the word co-occurrence network of the document, following (Tixier, Malliaros, and Vazirgiannis 2016). Hence, influential spreader detection comes down to keyword extraction. We tested whether the scores returned by P&C allowed to improve keyword extraction performance compared to using the scores computed on the original networks.

For cu and cw, we retained as keywords the words belonging to the main core of the network. Again, since we combine scores by taking the mean, the P&C scores may be floats. Thus, we rounded them up to the nearest integer before extracting the main core. For pr, we extracted the top 33% nodes as keywords. We tried with the following parameter values: $\varepsilon_a : \{0, 0.1, 0.2, 0.3\}$, $\varepsilon_d : \{0, 0.1, 0.2, 0.3\}$, $M : \{8, 32, 96\}$, $\mathbb{G} : \{ER, CL\}$ and $\delta_w : \{0, 1\}$. Excluding the cases where $\varepsilon_a = \varepsilon_d = 0$, this made for 180 combinations.

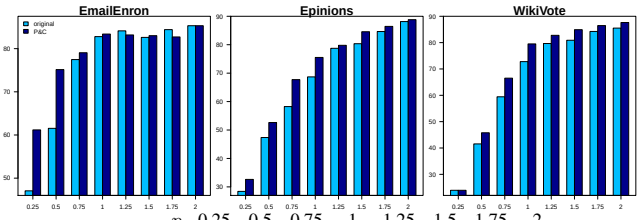
Results

Social networks

As can be seen from Table 2, for all vertex scoring functions and networks, using the combined scores returned by P&C to select the nodes from which to trigger the epidemic systematically leads to a greater severity. Moreover, the differences are substantial, comparable with the improvements reported in previous research, e.g., between k-truss and k-core (Malliaros, Rossi, and Vazirgiannis 2016). Although these results are the ones obtained with the combination of parameters that maximized the total number of nodes infected during the entire epidemic, one should note that the top nodes in terms of P&C scores are better spreaders than the top nodes in terms of original scores even during the early stages of the diffusion process.

Table 2: Social network results, for cu, cw, and pr (top to bottom). Numbers represent the average number of nodes infected during the SIR epidemic. +% is the percent severity increase.

		Time Steps						
Network	Scores	2	4	6	8	10	Total	+%
ENRON	**P&C**	**16**	**89**	**300**	**419**	**269**	**2,538**	3.76
	original	14	77	269	401	275	2,446	
EPINIONS	**P&C**	**8**	**34**	**110**	**245**	**317**	**2,436**	4.35
	original	7	30	100	224	301	2,330	
WIKIVOTE	**P&C**	**3**	**8**	**17**	**29**	**40**	**490**	3.47
	original	3	8	16	28	37	473	
ENRON	**P&C**	**26**	**141**	**407**	**445**	**226**	**2,724**	3.52
	original	20	110	345	433	253	2,628	
EPINIONS	**P&C**	**11**	**46**	**146**	**302**	**353**	**2,689**	2.42
	original	11	42	135	286	345	2,624	
WIKIVOTE	**P&C**	**5**	**12**	**24**	**39**	**50**	**612**	19.3
	original	4	9	18	31	42	513	
ENRON	**P&C**	**16**	**86**	**278**	**389**	**266**	**2,454**	4.93
	original	15	80	259	366	255	2,333	
EPINIONS	**P&C**	**11**	**42**	**132**	**276**	**336**	**2,598**	2.04
	original	11	41	127	267	326	2,545	
WIKIVOTE	**P&C**	**5**	**11**	**22**	**38**	**49**	**596**	2.35
	original	5	11	22	36	48	582	



p	0.25	0.5	0.75	1	1.25	1.5	1.75	2
ENRON	85	169	253	337	422	506	590	674
WIKIVOTE	71	142	212	283	354	424	495	566
EPINIONS	95	190	285	380	475	570	664	759

Figure 3: Fraction of the p% best spreaders (y axis) contained in the top p% nodes (x axis) in terms of P&C and original scores, for cu. The table shows the number of best spreaders for each p.

Finally, as shown in Figure 3, the P&C ranking is of better quality than the ranking provided by the original scores, especially for the very top nodes. On ENRON for instance, the top 0.5% (169) nodes according to P&C contain 75.15% of the 0.5% best spreaders (in terms of SIR), compared to only 61.54% when using the original scores. It means that the P&C scores place *more of the most influential spreaders* at the very top of the ranking than the original scores. This is a very valuable property, especially in practice when the end user can only select a very small number of nodes. E.g., there is often only budget to give away a tiny number of free samples in growth hacking and viral marketing.

Word co-occurrence networks

Performance is reported in Table 3 in terms of the usual information retrieval metrics: precision, recall, and F1-score. Precision measures how many of the nodes detected as influential are indeed influential (i.e., are keywords), while recall measures how many of the influential nodes were detected. The F1-score is the harmonic mean of precision and recall.

Table 3: Word networks results (macro-averaged scores).

Function	Scores	Precision	Recall	F1-score	+%
cu	**P&C**	**52.09**	**51.25**	**54.88**	5.70
	original	48.76	46.90	51.75	
cw	**P&C**	**50.53**	**48.54**	**52.50**	7.45
	original	48.07	46.81	48.86	
pr	**P&C**	**45.53**	**42.73**	**46.75**	2.33
	original	45.21	41.89	45.66	

Like on social networks, using the P&C scores in lieu of the original scores greatly improves performance for every node scoring function, with large absolute gains ranging from 1.09 to 3.64 in F1-score. Even though looking at relative improvements is sufficient to show that our P&C strategy is effective, it should be noted that the absolute scores we reach are equivalent to or exceed the state-of-the-art in unsupervised keyword extraction (Tixier, Malliaros, and Vazirgiannis 2016; Rousseau and Vazirgiannis 2015).

Discussion

Importance of parameters. As can be seen in Figure 4, most (but not all) parameter combinations return scores that allow the identification of better spreaders than the original scores. This suggests that while P&C is relatively robust to the choice of parameter values, some optimization is necessary to get the most out of the procedure. Tables 4 and 5 support this claim, by clearly demonstrating that there is no single best combination of parameters across networks and vertex scoring functions.

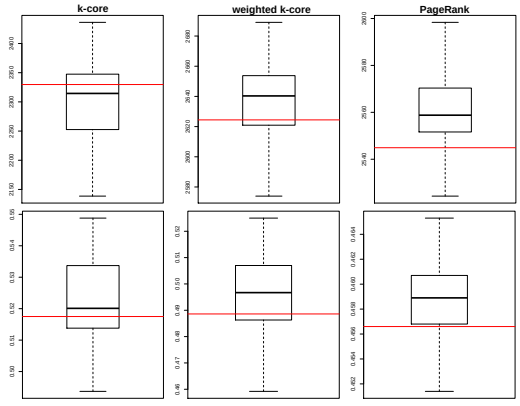


Figure 4: Top: distribution of epidemic severity for the 120 P&C parameter combinations tried on EPINIONS. Bottom: distribution of macro-averaged F1-score for the 180 combinations tried on Hulth2003. Red thin bars indicate performance reached using original scores.

Nonetheless, depending on the graph and/or scoring function, some parameters seem more important than others. For instance, for cu, selecting edges uniformly at random (ER model) during perturbation tends to work better than selecting edges in a biased way (CL model), which is consistent with the fact that cu ignores edge weights, unlike cw and pr. For cw, generating many perturbed versions of the original network (large values of M) also seems advantageous. Further research and analysis should help us understand what are the crucial parameters for different settings (graph type, size, density, diameter, scoring function...) and what are good priors for them, reducing the need for parameter tuning.

Table 4: Top 5 P&C parameter combinations, word networks (F1-scores).

δ_w	$\mathbb{G}$	M	ε_d	ε_a	**cu**	cw	pr
1	ER	8	0.1	0.3	**54.88**	50.67	45.99
0	ER	96	0.0	0.3	**54.85**	52.50	45.99
1	ER	96	0.3	0.3	**54.78**	51.76	45.65
0	ER	8	0.1	0.3	**54.69**	52.38	46.06
0	ER	32	0.0	0.3	**54.67**	52.31	45.90

δ_w	$\mathbb{G}$	M	ε_d	ε_a	cu	**cw**	pr
0	ER	96	0.0	0.3	54.85	**52.50**	45.99
0	ER	8	0.0	0.3	54.0.2	**52.38**	45.52
0	ER	8	0.1	0.3	54.69	**52.38**	46.06
0	ER	32	0.0	0.3	54.67	**52.31**	45.90
0	ER	96	0.1	0.3	54.62	**51.78**	46.20

δ_w	$\mathbb{G}$	M	ε_d	ε_a	cu	cw	**pr**
1	CL	32	0.0	0.3	50.94	45.33	**46.75**
0	CL	32	0.3	0.3	50.45	49.46	**46.57**
1	CL	8	0.0	0.3	51.86	46.99	**46.53**
0	CL	32	0.0	0.3	52.39	50.07	**46.47**
0	CL	32	0.2	0.3	52.22	50.49	**46.45**

Table 5: Top 5 P&C parameter combinations for ENRON, EPINIONS, WIKIVOTE (top to bottom) and pr, cw, cu (left to right). Scores indicate average epidemic severity.

δ_w	$\mathbb{G}$	M	ε_d	ε_a	pr
0	ER	64	0.2	0.2	2454
1	ER	16	0.2	0.2	2454
0	ER	16	0.05	0.2	2447
1	ER	64	0.2	0.2	2446
1	ER	16	0.05	0.2	2445
1	ER	16	0.2	0.2	2598
1	CL	64	0.2	0.05	2597
1	ER	64	0.2	0.2	2596
1	ER	16	0.1	0.2	2596
1	CL	64	0.2	0.2	2593
0	ER	16	0.2	0.2	596
0	ER	16	0	0.2	595
1	ER	64	0.05	0.2	594
0	ER	16	0.1	0.2	593
0	ER	16	0.2	0.05	593

δ_w	$\mathbb{G}$	M	ε_d	ε_a	cw
0	CL	64	0.2	0	2724
0	CL	64	0.1	0.1	2718
1	CL	64	0.2	0.2	2717
1	CL	64	0.1	0.1	2710
1	ER	64	0.05	0	2709
0	ER	64	0.1	0	2689
1	CL	64	0.05	0.05	2673
0	ER	64	0.2	0.05	2673
0	ER	64	0.1	0.2	2672
1	CL	64	0.1	0.1	2672
0	ER	64	0.2	0	612
0	CL	64	0.2	0.05	600
0	CL	64	0.2	0.1	589
0	CL	64	0.1	0.1	582
0	CL	64	0.1	0.05	578

δ_w	$\mathbb{G}$	M	ε_d	ε_a	cu
1	ER	64	0.2	0.05	2538
1	ER	64	0.2	0.1	2529
0	ER	64	0.1	0.1	2521
0	ER	64	0.05	0.05	2513
0	ER	64	0.1	0.05	2508
1	ER	16	0.2	0.05	2436
1	ER	16	0.2	0.1	2436
1	CL	64	0.2	0	2423
1	ER	16	0.2	0	2419
1	ER	16	0.2	0.2	2391
1	CL	16	0.05	0.05	490
1	ER	64	0.2	0	488
1	CL	16	0.1	0.05	487
1	ER	16	0.1	0	487
1	ER	16	0.05	0	485

P&C improves even the performance of PageRank. This was unexpected, as PageRank is believed to be stable to edge-based perturbations (Ipsen and Wills 2006; Ng, Zheng, and Jordan 2001). The implication could be that our P&C procedure is beneficial to any node scoring function, not only unstable ones, or that PageRank features some level of instability. While we provide some evidence supporting the former implication in the Theoretical Analysis section, the latter is a legitimate possibility as well. It has indeed been suggested that the stability of PageRank depends on the network topology. For instance, PageRank is much more stable for scale-free graphs like the Web than for random networks (Ghoshal and Barabási 2011).

For word networks, adding edges is beneficial. Interestingly, for word co-occurrence networks, adding edges or incrementing the weights of already existing edges seems much more important than deleting edges ($\varepsilon_a \geq \varepsilon_d$), regardless of the scoring function. This is equivalent to copying and pasting words from/to the input text, which can be seen as a form of *data augmentation*. It could also be interpreted as having a sliding window of stochastic size featuring an additional *masking* mechanism such that edges are drawn between a subset only of the words in each instantiation of the window. Data augmentation and stochastic windows were proven very beneficial in Computer Vision and NLP (Krizhevsky, Sutskever, and Hinton 2012; Mikolov et al. 2013), so this could explain why $\varepsilon_a \geq \varepsilon_d$ works well with word networks.

Next, we backup our positive empirical results by explaining why P&C is effective from a theoretical perspective.

Theoretical Analysis

Underlying graph. Let us assume the existence of a true but unavailable underlying graph G^*, of which the available graph G is a snapshot, or sample, so that G features the same nodes as G^* but has a slightly different edge set. This is analogous to the traditional assumption made in statistics that *a given dataset represents a sample of a true but unknown distribution*. Since G^* is unavailable, we have to find a way to emulate sampling from G^* by using only G. One solution is to perturb G.

True ranking. Let us also assume the existence of a true ranking $R = \{l(1), \ldots, l(n)\}$ of the nodes $\{v_1, \ldots, v_n\}$ of G, that associates each node v_i with one of K labels $l(i)$, where v_i is ranked before v_j if $l(i) > l(j)$. $K \leq n$ as some nodes may have equivalent spreading capabilities. This true ranking can be a given, or can be computed, for instance with the SIR model.

Objective. Let $s : V \mapsto \mathbb{R}^{|V|}$ be a vertex scoring function, i.e., a function that associates each node of G with a real number, and let $\hat{R}$ be the ranking induced by s on the nodes of G. $\hat{R}$ can be seen as an *estimate* of the true ranking R, and s as an *estimator*. Let us also assume that the quality of the estimate provided by s (goodness of fit) is measured by a metric MET accepting $\hat{R}$ and R as input and taking values in $[0, 1]$. The objective of s is to maximize MET. MET is a random variable (RV) as $\hat{R}$ is a RV.

Perturbation as sampling. In each of the M edge-perturbed version $\tilde{G}_m$ of G, the individual node scores, and by extension the rankings $\hat{R}_m$, randomly vary, as our perturbation strategy is stochastic. We can thus consider the $\hat{R}_m$ to be RVs. Moreover, since the $\tilde{G}_m$ are generated independently, the $\hat{R}_m$ are independent. Therefore, perturbing G is akin to *sampling independent realizations from the true underlying graph G^**.

Definitions: bias and variance of a vertex scoring function. Our goal is to study the impact of P&C on the goodness of fit of s. In regression, the error is traditionally decomposed into bias and variance terms. We adopt this framework and define in what follows the bias and variance of s.

In the regression setting, $y = f(x) + \epsilon$, $\sigma^2 = \mathrm{var}[\epsilon]$, $\hat{f}$ is an estimator of f, and we have the following well-known breakdown of expected squared error of the estimation into (squared) bias, variance, and irreducible error terms:

$$\mathbb{E}[(y - \hat{f}(x))^2] = \mathrm{bias}[\hat{f}(x)]^2 + \mathrm{var}[\hat{f}(x)] + \sigma^2 \quad (2)$$

$$\mathrm{bias}[\hat{f}(x)] = \mathbb{E}[\hat{f}(x) - f(x)] \quad (3)$$

$$\mathrm{var}[\hat{f}(x)] = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2] \quad (4)$$

The expectation is computed for different samples drawn from the same underlying distribution. By analogy, in our setting, we can define the bias and variance of s as:

$$\mathrm{bias}[s] = \mathbb{E}[1 - \mathrm{MET}] = 1 - \mathbb{E}[\mathrm{MET}] \quad (5)$$

$$\mathrm{var}[s] = \mathbb{E}[(\mathrm{MET} - \mathbb{E}[\mathrm{MET}])^2]. \quad (6)$$

The bias captures, on average, how close the estimated ranking $\hat{R}$ provided by s is to the true ranking R (for which MET is equal to 1), while the variance measures the instability of $\hat{R}$ (variability around its mean). The expectation is to be understood as computed over a set of observations of G^*.

Since $\forall$ RVs X, Y and $k \in \mathbb{R}$, $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$, $\mathbb{E}[k] = k$, and $\mathbb{E}[kX] = k\mathbb{E}[X]$, developing Eq. 6 gives:

$$\mathrm{var}[s] = \mathbb{E}[\mathrm{MET}^2 - 2\mathbb{E}[\mathrm{MET}]\mathrm{MET} + \mathbb{E}^2[\mathrm{MET}]] \quad (7)$$

$$\mathrm{var}[s] = \mathbb{E}[\mathrm{MET}^2] - 2\mathbb{E}^2[\mathrm{MET}] + \mathbb{E}^2[\mathrm{MET}]. \quad (8)$$

Summing the squared bias and variance terms thus gives:

$$\mathrm{bias}^2[s] + \mathrm{var}[s] = 1 - 2\mathbb{E}[\mathrm{MET}] + \mathbb{E}[\mathrm{MET}^2] \quad (9)$$

$$\mathrm{bias}^2[s] + \mathrm{var}[s] = \mathbb{E}[(\mathrm{MET} - 1)^2] \quad (10)$$

which can be interpreted as the expectation of the squared error, like in the case of regression.

Theorem: P&C reduces error.

Proof. Recall that the P&C score s_{pc} of node v_i is defined as the average of the scores its gets in each of the M perturbed graphs $\{\tilde{G}\} = \{\tilde{G}_m\}_{m=1}^M$ generated from G:

$$s_{\mathrm{pc}}(v_i) = \frac{1}{M} \sum_{m=1}^{M} s_{\tilde{G}_m}(v_i). \quad (11)$$

We can write:

$$\hat{R}_{\mathrm{pc}} = \mathbb{E}_{\{\tilde{G}\}}[\{\hat{R}\}] \quad (12)$$

where $\{\hat{R}\} = \{\hat{R}_m\}_{m=1}^M$. This means that the P&C estimate $\hat{R}_{\mathrm{pc}}$ of the true ranking R is the average of the estimates $\hat{R}_m$ over the M perturbed graphs. Similarly, the goodness of fit of the P&C ranking can be written:

$$\mathrm{MET}_{\mathrm{pc}} = \mathbb{E}_{\{\tilde{G}\}}[\mathrm{MET}(\{\hat{R}\}, R)]. \quad (13)$$

Thus, evaluating Eq. 9 over $\{\tilde{G}\}$, and using Eq. 13 above:

$$\mathbb{E}_{\{\tilde{G}\}}[(\mathrm{MET} - 1)^2] = 1 - 2\mathbb{E}_{\{\tilde{G}\}}[\mathrm{MET}] + \mathbb{E}_{\{\tilde{G}\}}[\mathrm{MET}^2] \quad (14)$$

$$\mathbb{E}_{\{\tilde{G}\}}[(\mathrm{MET} - 1)^2] = 1 - 2\mathrm{MET}_{\mathrm{pc}} + \mathbb{E}_{\{\tilde{G}\}}[\mathrm{MET}^2] \quad (15)$$

where $\mathrm{MET}(\{\hat{R}\}, R)$ is simply written MET for readability. Plus, since $\forall$ RV X and $k \in \mathbb{R}$, $\mathbb{E}^2[X] \geq \mathbb{E}[X^2]$ and $\mathbb{E}[k] = k$, using again Eq. 13, and since $\mathbb{E}$ is monotone:

$$\mathbb{E}_{\{\tilde{G}\}}[(\mathrm{MET} - 1)^2] \geq 1 - 2\mathrm{MET}_{\mathrm{pc}} + \mathbb{E}^2_{\{\tilde{G}\}}[\mathrm{MET}] \quad (16)$$

$$\geq (1 - \mathrm{MET}_{\mathrm{pc}})^2 \quad (17)$$

$$\mathbb{E}_{\{\tilde{G}\}}[(\mathrm{MET} - 1)^2] \geq \mathbb{E}_{\{\tilde{G}\}}[(1 - \mathrm{MET}_{\mathrm{pc}})^2]. \quad (18)$$

Ineq. 18 shows that ***the mean squared error of P&C (RHS) is always lower than or equal to the original mean squared error (LHS)***, which is an important result. The improvement can come from reducing bias, variance, or both. □

Sample bias and variance. To understand how Ineq. 18 holds in practice, we randomly selected 16 Hulth2003 word networks, and generated 50 perturbed version of each. As

previously explained, this can be considered as drawing 50 independent realizations from the underlying graph that generated each network. With the cu function, we then scored the nodes of each graph in the sample with and without using our P&C strategy, and computed the goodness of fit of each ranking, using the Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen 2002) as the metric MET. The NDCG is a standard metric for assessing ranking quality in Information Retrieval (IR). In our case, the more of the most influential spreaders are placed on top of $\hat{R}$, the better the NDCG. More precisely, NDCG is computed as:

$$\mathrm{NDCG} = {}^{\mathrm{DCG}}/_{\mathrm{IDCG}} \tag{19}$$

where DCG is the Discounted Cumulative Gain computed on $\hat{R}$ and IDCG is the ideal DCG computed on R. NDCG is maximal and equal to 1 if $\hat{R}$ matches R exactly. Generally in IR, the DCG is computed over a shortlist of the best results, but we can assume without loss of generality that it is computed over the full list of n nodes:

$$\mathrm{DCG} = \sum_{i=1}^{n} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{20}$$

where i designates the rank of node v_i in the list. We used as the relevance score rel_i of v_i its SIR influence, i.e., the average number of nodes infected at the end of multiple epidemics triggered from it. We finally computed bias and variance from the set of 50 NDCGs by using Eq. 5 and Eq. 6. We repeated the same procedure for the WIKIVOTE network.

Results are shown in Table 6. As can be seen for word networks, P&C reduces both bias and variance, although its major contribution appears to lie in the consistent reduction of bias. The average of the averages of the NDCGs is 0.58 for the rankings obtained with the original scores and 0.73 for P&C, which means that the rankings returned by the P&C scores fit the true rankings much better. On WIKIVOTE, P&C reduces bias, but not variance, which is initially very low. The P&C NDCG is 0.2032 while the original is 0.1892, indicating again that P&C returns better rankings.

Table 6: Top: sample bias ($\times 10^2$) and variance ($\times 10^3$) for 16 randomly selected word networks. Bottom: Sample bias ($\times 10^2$) and variance ($\times 10^5$) for WIKIVOTE. Lower is better.

	Original	P&C	
bias	37.06	24.92	↘
var	4.78	1.17	↘
bias	41.92	16.66	↘
var	12.83	4.72	↘
bias	48.44	35.64	↘
var	6.02	5.43	↘
bias	42.86	30.23	↘
var	0.08	4.44	
bias	45.16	31.23	↘
var	0.03	4.09	
bias	64.56	38.97	↘
var	1.87	4.04	
bias	24.75	14.05	↘
var	0.58	2.52	
bias	31.72	19.00	↘
var	0.58	6.78	

	Original	P&C	
bias	51.59	34.97	↘
var	0.66	1.16	
bias	39.16	27.22	↘
var	0.04	1.79	
bias	66.43	54.66	↘
var	0.09	0.76	
bias	46.97	21.98	↘
var	1.54	2.71	
bias	26.40	20.75	↘
var	0.04	0.49	
bias	37.29	24.40	↘
var	0.27	2.17	
bias	33.76	18.68	↘
var	0.18	2.50	
bias	41.61	24.12	↘
var	6.85	6.60	↘

	Original	P&C	
bias	81.08	79.68	↘
var	0.00	0.64	

P&C for networks differs from bagging. Our P&C strategy reducing mainly bias rather than variance suggests that it differs from the most famous of the P&C strategies, bootstrap aggregation (bagging). Indeed, bagging can increase variance when it fails (Grandvalet 2004), but it is widely accepted that it cannot significantly reduce bias. Actually, as shown by (Breiman 1996a), bagging is only effective through reducing the high variance of unstable learners (e.g., fully-grown decision trees), and does not work well with stable algorithms (e.g., nearest-neighbors approaches) because it cannot reduce bias. Another obvious difference is that a bootstrap sample has always the same size as the original dataset, whereas in our case, a perturbed graph has as many edges as the original graph only when $\varepsilon_a = \varepsilon_d$.
To strictly emulate bagging, we would need to adopt a different framework in which we would draw *edges* from a true underlying distribution rather than graphs. The perturbation step would only consist in sampling edges with replacement (i.e., bootstrapping) from the original set of edges. Some edges would be selected more than once (their weights would be incremented), while some edges would not be selected at all. In the final perturbed network, 63.2% of unique edges would carry over from the original network, but no new edge would be present. This would remove the need for the ε_a and ε_d parameters, at the cost of losing flexibility. The improvement brought by P&C would be obtained mainly by reducing variance as:

$$\begin{aligned} \mathrm{var}\big[\hat{R}_{\mathrm{pc}}\big] &= \mathrm{var}\left[\frac{1}{M}\sum_{m=1}^{M}\hat{R}_m\right] \\ &= \frac{1}{M^2}\sum_{m=1}^{M}\mathrm{var}\big[\hat{R}_m\big]. \end{aligned} \tag{21}$$

The extent to which error would be reduced would thus only depend on the amount of uncorrelation among the rankings $\hat{R}_m$, as correlation adds positive covariance terms to the RHS of Eq. 21, i.e., increases variance. In other words, if our approach was equivalent to bagging, it would only work for unstable vertex scoring functions.

The fact that our procedure is effective even for PageRank, which is considered relatively stable, corroborates the empirical findings that our method is capable of reducing bias in addition to variance. Rather than bagging, we think that our approach is more closely related to *noise injection* techniques such as the *noisy* and *smoothed* bootstrap (Raviv and Intrator 1996; Silverman 1986), and, as already mentioned, to *data augmentation* strategies. More generally, perturbing graphs could also be seen as a form of *adversarial training* (Zügner, Akbarnejad, and Günnemann 2018).

Conclusion

We proposed what is, to the best of our knowledge, the first application of the Perturb and Combine (P&C) strategy to graphs. Experiments on various real-world networks demonstrate that for all vertex scoring functions tested, the P&C scores allow to identify better spreaders than the scores computed on the original networks. Furthermore, the P&C scores can be obtained at little extra cost through parallelization. We explain our positive empirical results through a theoretical analysis.

References

Adiga, A., and Vullikanti, A. K. S. 2013. How robust is the core of a network? In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 541–556. Springer.

Batagelj, V., and Zaveršnik, M. 2002. Generalized cores. *arXiv preprint cs/0202039*.

Breiman, L. 1996a. Bagging predictors. *Machine learning* 24(2):123–140.

Breiman, L. 1996b. Bias, variance, and arcing classifiers.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Chakrabarti, D.; Wang, Y.; Wang, C.; Leskovec, J.; and Faloutsos, C. 2008. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)* 10(4):1:1–1:26.

Chung, F., and Lu, L. 2002. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences* 99(25):15879–15882.

Cohen, J. 2008. Trusses: Cohesive subgraphs for social network analysis. *National Security Agency Technical Report* 16.

Erdös, P., and Rényi, A. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5:17–61.

Ghoshal, G., and Barabási, A.-L. 2011. Ranking stability and super-stable nodes in complex networks. *Nature communications* 2:394.

Goltsev, A. V.; Dorogovtsev, S. N.; and Mendes, J. F. F. 2006. k-core (bootstrap) percolation on complex networks: Critical phenomena and nonlocal effects. *Physical Review E* 73(5):056101.

Govindan, P.; Wang, C.; Xu, C.; Duan, H.; and Soundarajan, S. 2017. The k-peak decomposition: Mapping the global structure of graphs. In *Proceedings of the 26th International Conference on World Wide Web*, 1441–1450. International World Wide Web Conferences Steering Committee.

Grandvalet, Y. 2004. Bagging equalizes influence. *Machine Learning* 55:251–270.

Hoppensteadt, F. 1975. *Mathematical Theories of Populations: Deomgraphics, Genetics, and Epidemics*, volume 20. Siam.

Hulth, A. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 216–223. Association for Computational Linguistics.

Ipsen, I. C., and Wills, R. S. 2006. Mathematical properties and analysis of googles pagerank. *Bol. Soc. Esp. Mat. Apl* 34:191–196.

Järvelin, K., and Kekäläinen, J. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)* 20(4):422–446.

Kermack, W. O., and McKendrick, A. G. 1932. Contributions to the mathematical theory of epidemics. ii. the problem of endemicity. volume 138, 55–83. JSTOR.

Kitsak, M.; Gallos, L. K.; Havlin, S.; Liljeros, F.; Muchnik, L.; Stanley, H. E.; and Makse, H. A. 2010. Identification of influential spreaders in complex networks. *Nature physics* 6(11):888–893.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Leskovec, J.; Adamic, L. A.; and Huberman, B. A. 2007. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)* 1(1):5.

Leskovec, J., and Krevl, A. 2014. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Malliaros, F. D.; Rossi, M.-E. G.; and Vazirgiannis, M. 2016. Locating influential nodes in complex networks. *Scientific reports* 6:19307.

Mihalcea, R., and Tarau, P. 2004. TextRank: bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ng, A. Y.; Zheng, A. X.; and Jordan, M. I. 2001. Link analysis, eigenvectors and stability. In *International Joint Conference on Artificial Intelligence*, volume 17, 903–910. LAWRENCE ERLBAUM ASSOCIATES LTD.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Raviv, Y., and Intrator, N. 1996. Bootstrapping with noise: An effective regularization technique. *Connection Science* 8(3-4):355–372.

Rousseau, F., and Vazirgiannis, M. 2015. Main core retention on graph-of-words for single-document keyword extraction. In *European Conference on Information Retrieval*, 382–393. Springer.

Sariyuce, A. E.; Seshadhri, C.; Pinar, A.; and Catalyurek, U. V. 2015. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings of the 24th International Conference on World Wide Web*, 927–937. International World Wide Web Conferences Steering Committee.

Seidman, S. B. 1983. Network structure and minimum degree. *Social networks* 5(3):269–287.

Silverman, B. W. 1986. *Density estimation for statistics and data analysis*. Routledge.

Tixier, A.; Malliaros, F.; and Vazirgiannis, M. 2016. A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1860–1870.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial attacks on neural networks for graph data. *arXiv preprint arXiv:1805.07984*.