



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Master's Thesis

Scalable and Parallelizable Influence
Maximization with Random Walk Ranking
and Rank Merge Pruning

Seungkeol Kim (김 승 결)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2015



랜덤 워크와 랭크 병합 프루닝을 적용한
대규모 그래프 상에서의 영향력 최대화

Scalable and Parallelizable Influence
Maximization with Random Walk Ranking
and Rank Merge Pruning



Scalable and Parallelizable Influence
Maximization with Random Walk Ranking
and Rank Merge Pruning

by
Seungkeol Kim

Department of Computer Science and Engineering
Pohang University of Science and Technology

A thesis submitted to the faculty of Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Department of Computer Science and Engineering

Pohang, Korea

12. 23. 2014

Approved by



Major Advisor



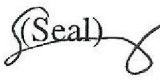


Scalable and Parallelizable Influence
Maximization with Random Walk Ranking
and Rank Merge Pruning

Seungkeol Kim

The undersigned have examined this thesis
and hereby certify that it is worthy of acceptance for a master's
degree from POSTECH

12/23/2014

Committee	Chair	Hwanjo Yu	(Seal) 
	Member	Jeong-Hyon Hwang	(Seal) 
	Member	Seung-Won Hwang	(Seal) 



M C S E 김 승 결 Seungkeol Kim

20110724 Scalable and Parallelizable Influence Maximization with Random Walk Ranking and Rank Merge Pruning, 랜덤 워크와 랭크 병합 프루닝을 적용한 대규모 그래프 상에서의 영향력 최대화
Department of Computer Science and Engineering, 2015, 44P,
Advisor : Hwanjo Yu. Text in English.

ABSTRACT

Abstract

Influence maximization problem is to find the k most influential nodes(or individuals) on a social network G that maximize the influence spread with an underlying influence diffusion model. As the social network services take a large portion of modern life, influence maximization became an uprising important problem in viral marketing and many methods have been developed. Previous methods, however, commonly suffer from very low evaluation of influence spread. They use greedy approximation to deal with the NP-hardness of selecting k seed nodes, and to run the greedy method, they spend significant amount of time evaluating the influence spread of every individual nodes; which takes a big portion of the total execution time for the influence maximization problem.

In this paper, we propose an effective pruning method based on Random Walk and Rank Merge, which prunes out uninfluential nodes effectively and dramatically reduces the number of influence evaluations at the initial step. Our pruning method combined with IPA algorithm significantly reduces the total execution time of the algorithm by up to ten times, which becomes the fastest influence maximization



among all the state-of-the-art methods (i.e. IPA [10], IRIE [8] and TIM⁺ [9]). Additionally, our method is easily parallelizable with few lines of OpenMP statements, and the parallel version of our method further speeds up the execution time by up to 5 times with 6-core CPU.



Contents

1	Introduction	1
2	Preliminaries and Related Works	5
2.1	Notations	5
2.2	Problem Definition	5
2.3	Independent Cascade model	7
2.4	Assigning Edge Weight	8
2.5	Greedy Framework	9
2.5.1	Kempe et al.'s approach	9
2.5.2	CELF greedy algorithm	10
2.6	Independent Path Algorithm	12
2.6.1	Other Existing Works	15
3	Proposed Algorithm	16
3.1	Random Walk Pruning	16
3.2	RWP-IPA	22
4	Experiments	25
4.1	Experiment Setting	25
4.1.1	Running Environment	25
4.1.2	Datasets	25
4.1.3	Algorithms	26
4.1.4	Parameter Setting	27
4.2	Experiment Results	28



4.2.1	Effectiveness of RWP	28
4.2.2	Processing Time (IPA vs RWP-IPA)	30
4.2.3	Processing Time (All Algorithms)	31
4.2.4	Influence Spread	32
4.2.5	Memory Usage	33
4.2.6	Parallelization Effect	34
5	Conclusion	39



List of Figures

1	Sample graph and path collection starting from a	13
2	An example of selecting the number of iteration steps and required k' respectively in Random Walk Pruning.	19
3	Average coverage of seed nodes of IPA varying the size of k' in each iteration.	35
4	Running time comparison of IPA and RWP-IPA	36
5	Processing time of each algorithm in five datasets.	36
6	Average coverage of seed nodes of IPA varying the size of k' in each iteration.	37
7	Speed-up of parallel RWP-IPA.	38



List of Tables

1	Frequently used notations.	6
2	Effectiveness of RWP in Stanford dataset	21
3	Processing time of IPA and RWP-IPA on DBLP dataset	23
4	Statistics of the datasets	26
5	Selected parameters for Greedy, PMIA, IPA, TIM ⁺ and RWP-IPA	28
6	Required S' to select all the seed nodes of IPA	29
7	Generated S' after applying RWP with $z = 5$ and $k' = 3k = 150$	30
8	Time consumption of each phase in IPA algorithm	30
9	Memory usage of for for PMIA, IPA, TIM ⁺ and RWP-IPA	33



1 Introduction

Nowadays, social networks such as blogs, Facebook and Twitter take a large portion of human life. People express their opinions on social network and share their information across the world beyond the border. As the information spreads across the social network people in social networks are influenced by the others and such influence spread is called "*word-of-mouth*" effect [1]. These phenomenon takes a important role in *viral marketing* and finally P. Domingos and M. Richardson introduced influence maximization problem [2, 3].

While [2, 3] introduced the influence maximization problem in a probabilistic perspectives, Kempe et al. quantified the influence spread and make influence maximization problem as a discrete optimization [4]. Kempe et al. introduced several influence diffusion models and invented greedy approximation algorithm which guarantees $(1 - 1/e)$ -approximation of optimal solution. Their research and assumptions becomes a basis of influence maximization problem and many researches are following in database and data mining communities [1, 5, 7, 6, 8, 11, 9, 10].

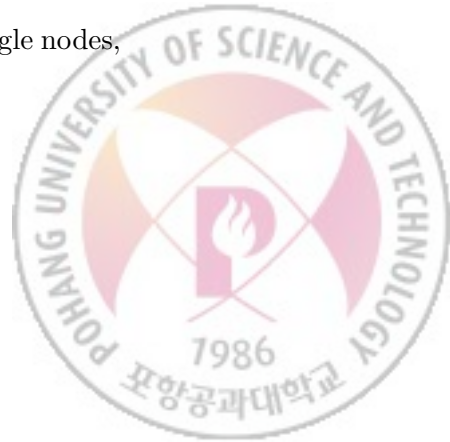
In the view of discrete optimization, influence maximization problem requires a social network graph and influence diffusion model. A social network is given as a abstracted weighted directed graph $G(V, E, W)$ where nodes V are individuals or communities, edges E are interaction between nodes and weights W are probability assigned to each edges that a node influences the other node along the edge. Influence diffusion model describes the aspect how the influence is propagated along the graph from the seed nodes to the



other nodes and the independent cascade model (IC model) is widely used in research communities. Finally with the given social network and influence diffusion model and given number k , influence maximization problem is to find the k influential nodes that maximize the influence spread in the graph $G(V, E, W)$ [4].

Even Kempe et al.'s formulated the problem into discrete way, still influence maximization on large-scaled social network is challenging. Selecting the k optimal seed nodes is NP-hard [4] and evaluating the influence spread of selected seed candidates is #P-hard at once [7]. To overcome NP-hardness, [4] introduced greedy approximation with $(1-1/e)$ -approximation of the optimal solution. Also many works challenged on #P-hardness in different way. To boost a single influence evaluation, Wei et al. used degree and local structures to evaluate influence spread of the nodes [6, 7]; Kyomin et al used belief propagation to initialize influence of every nodes in the graph [8]; Brogs et al introduced sampling method with theoretical support [11], and Tang et al. improved [11] with several optimization tweaks [9]; Kim et al. used paths between nodes as influence evaluation unit [10]. To reduce the number of influence evaluations, Leskovec et al. invented CELF greedy algorithm using lazy-forward optimization [5].

Kim et al. invent the IPA algorithm which propose a simple and fast approximation of influence spread under the IC model [10] using independent influence as a influence evaluation unit. The IPA algorithm has three phases, traversing, re-organizing, and updating phase which corresponds to traverse influence paths and evaluating influence spread of every single nodes,



distribute influence paths for updating phase, and pick up k seed nodes and update the change of marginal influence by new seed nodes. While the serial version of the IPA algorithm's updating phase runs in $O(k|O_{S \cup \{u\}}|n_{uv})$ time which shows aggressively fast seed selection, traversing and re-organizing phase takes both $O(m|O_{avg}|n_{uv})$ which is quite time consuming process. This phenomenon is caused because the IPA algorithm traverse influence paths from every nodes in V which is inefficient.

In this paper, we propose the Random Walk Pruning (RWP) algorithm which prunes out uninfluential nodes effectively and dramatically reduce the number of influence evaluation at the initial step which is #P-hard per each evaluation. Then we propose RWP-IPA which integrated RWP into the IPA algorithm. RWP efficiently reduces the candidate nodes which influence spreads should be evaluated and dramatically decrease the time consumption of traversing and re-organizing phase of IPA. Also RWP is easily parallelizable with few lines of OpenMP [12] statements so that occurs greatest synergy with the parallelizable IPA algorithm. The final time complexity of the RWP-IPA is $O((z((m+n)/c + n \log n)) + ((zk'|O_{avg}|n_{uv})/c) + zk'|O_{avg}|n_{uv} + k(((|O_{S \cup \{u\}}|n_{uv})/c) + (c-1)))$, dominated by $O(z((m+n)/c + n \log n))$, in the worst case which is the most optimal time complexity researched in our best knowledge.

We performed extensive experiments on real world social network graphs which the biggest one has millions of nodes, and those experiments supports our following contributions:

- The RWP successfully removes uninfluential nodes and reduces the



number of influence evaluation required at the initial selection. In the biggest dataset we used, the number of influence evaluations in the RWP-IPA decreases to 0.003% comparing to that of the IPA algorithm.

- The RWP-IPA is always an order of magnitude faster than PMIA [7], and shows comparable computation time versus TIM⁺ [9], even TIM⁺ uses aggressive parameters and sacrifices the influence spread guarantee.
- The RWP-IPA accurately approximate the influence spread and shows no more than 10% difference of that of Greedy solution.
- The RWP-IPA is easily parallelizable and gets even more speed-up while the number of cpu cores used increases.



2 Preliminaries and Related Works

In this section, we revisit the Independent Cascade model (IC model) and the formulated definition of influence maximization problem [4] defined by Kempe et al. We also

2.1 Notations

To demonstrate our work easily, we show several frequently used notations here. Table 1 shows the notations that are frequently used. The definitions will be introduced once more when the notations first appear in the following of the paper.

2.2 Problem Definition

Previously the influence maximization problem was defined in probabilistic perspective [2, 3], Kempe et al. define the problem formally so that the problem can be solved in algorithmic way [4]. We begin the paper with revisiting the formalized definition of influence maximization problem.

To solve the influence maximization problem, we need a social network graph which the problem will handle. The social network graph is abstracted to $G(V, E, W)$ and given. The definition of abstracted social network graph is following:

Definition 1. *A abstracted social network graph is defined as $G(V, E, W)$. V represents the set of nodes in the graph. E represents the set of edges where $E = \{(u, v) | u, v \in V\}$. W is the influence propagation probability assigned to*



Table 1: Frequently used notations.

Notation	Definition
G	Social network graph
V, m	Nodes & number of nodes $ V $
E, n	Edges & number of edges $ E $
W	Weights assigned to each edges
D_{in}, D_{out}	In-degree, out-degree
E_{in}, E_{out}	Incoming, outgoing edges
k	The size of the seed set to be found
k'	Number of seed candidates in each RWP iterations
z	Number of RWP iterations
S, S'	Selected seeds, seed candidates
$p, ipp(p)$	Influence path and its influence propagation probability
$P_{A \rightarrow B}$	Influence path collection from A to B
O_A	Influenced area of A
$\sigma, \hat{\sigma}$	Influence spread, influence spread approximated by IPA
c	The number of CPU cores used
n_{uv}	Average size of the path collection $P_{u \rightarrow v}$
$ O_{avg} $	Average size of the influenced area O_u
$ p_{avg} $	average memory consumption per a influence path

the edges, $\forall (u, v) \in E, \exists w_{(u,v)} \in W, \forall w_{(u,v)} \in (0, 1]$.

The weights in W can be pre-given with the graph or it can be learnt with a special rule according to the graph topology $G(V, E)$.



When a social network graph $G(V, E, W)$ and specific number k is given, the influence maximization problem is defined as follows:

Definition 2. *The influence maximization problem is to find the seed node set S with size of k in the graph which*

$$S = \arg \max_{S' \subseteq V, |S'|=k} \sigma(S') \quad (1)$$

where $\sigma(S')$ is an influence quantifier that quantifying the influence spread on G by node set S' .

Now the influence maximization problem is formally defined and becomes a problem finding a combination of nodes. However, still its non-trivial that the possible combinations are ${}_nC_k$ which is NP-hard problem to be solved, and also evaluating the influence is intractable and unknown so that solving $\sigma(S')$ is #P-hard problem [4]. To overcome those challenges, [4] proposed greedy approximation which guarantees the $(1 - 1/e)$ ratio of the optimal solution for NP-hardness, Monte-Carlo simulation based influence evaluation in IC model case for #P-hardness.

2.3 Independent Cascade model

To evaluate the influence spread σ computationally by using monte-carlo simulation, we adopt the IC model as the influence diffusion model. In the real world social network, an information is uploaded by a user. If their friends see the information and influenced, then the neighbors like, retweet or share the information and the information is shown to the friend's neighbors. IC model reflects this phenomenon of real world that people are influenced by their neighbors directly man-to-man.



In the IC model, a node can have two states, active and inactive states. The seed nodes S are active, otherwise inactive in the initial step $t = 0$ and the activated nodes at $t = 0$ is $A_0=S$. In each step $t = n$, activated nodes at $t = n - 1$ have a single chance to activate their inactive out neighbors $\{v|u \in A_{n-1}, (u, v) \in E, v \notin A\}$ with probability $w(u, v)$ where $A = \cup_{t=0}^{n-1} A_t$. The influence propagation terminated when there is no activated in the previous step $A_n = \emptyset$. The total influence spread is $|A|$.

2.4 Assigning Edge Weight

Learning information propagation probabilities assigned to edges is another topic in data mining area. Saito et al. learn the propagation probabilities using the past propagation log targeting IC model [14]. Goyal et al. learnt the probabilities at the moment when a user is in an action [15], and obtained probabilities from the previous action log without simulating influence diffusion process.

Because we are not aiming to learn the influence probability from the graph, we use frequently used weighting polity by iteratures [4, 5, 7, 6, 8, 11, 9, 10], Weighted Cascade model (WC model), which the probability of an edge is defined by the in-degree of a node. Weight assigned to an edge (u, v) is assigned $w_{(u,v)} = 1/D_{in}(v)$



2.5 Greedy Framework

2.5.1 Kempe et al.'s approach

Finding the optimal solution for the influence maximization is a non-trivial challenge with great search space. To find the optimal k nodes with given graph G ($k \ll n$ in most cases), we need to evaluate the influences of all $nC_k \approx n^k$ cases. Kempe et al. prove a proof that finding the optimal k seed nodes in IC model is NP-hard problem so that it is impossible to compute in polynomial time [4]. To get through this problem, Kempe et al. propose a greedy approximation algorithm [4], selecting a new seed node u with the highest marginal influence gain which is,

$$u = \arg \max_{v \in V} (\sigma(S \cup \{v\}) - \sigma(S))$$

until $|S| = k$. The greedy algorithm guarantees $1 - 1/e$ of the optimal solution by using the properties of $\delta(S)$. The number of influence evaluations required is reduced to $\sum_{i=1}^k (n - i + 1)$ cases.

Definition 3. A function $f(\cdot)$ has non-negativity, monotonicity, and sub-modularity if and only if

For $S \subset S' \subset V$, $v \in V$,

$$\forall (S \subset V). f(S) \geq 0, \quad \text{non-negativity}$$

$$f(S) \leq f(S'), \quad \text{monotonicity}$$

$$f(S \cup \{v\}) - f(S) \geq f(S' \cup \{v\}) - f(S'). \quad \text{sub-modularity}$$

Theorem 1. The influence maximization problem of Definition 2 is NP-hard under IC model.



Proof. see Theorem 2.4 of [4]. □

Theorem 2. *The greedy solution of the influence maximization problem approximates the optimal solution with the ratio of $(1 - 1/e)$.*

Proof. see Theorem 2.1, 2.2 of [4]. □

The influence spread $\sigma(S)$ is evaluated by using the average of 20,000 Monte-Carlo simulations.

2.5.2 CELF greedy algorithm

Even Kempe et al's greedy algorithm needs $N - i + 1$ influence evaluations to add a seed node, still it is inefficient considering the heavy workload of Monte-Carlo simulations. Leskovec et al. propose CELF greedy algorithm [5] which dramatically reduces influence evaluations using lazy-forward optimization. From the sub-modularity of the $\sigma(S)$, Leskovec et al. observed that only a few influence evaluation is needed if we maintain the marginal influence gain in a priority queue.

The CELF greedy algorithm is reported in Algorithm 2.5.2. The CELF greedy algorithm constructs a priority queue PQ maintains the marginal influence spread increase of each nodes. When a node u is popped from PQ (line 9), it is checked whether it is updated just before or not (line 10) which means u has the highest marginal influence spread increase. If true, u is selected as a new seed node (line 11), otherwise new influence spread increase is evaluated (line 13) and update the PQ (line 14). Because of the



CELFgreedy(G, k)

Require: G : social network graph,

k : the number of seed nodes to be selected

Ensure: S : k influential seed nodes

```
1:  $S, PQ \leftarrow \emptyset$ 
2: for all  $v \in V$  do
3:   evaluate  $\sigma(\{v\})$ 
4:    $PQ.push(\sigma(\{v\}), v)$ 
5: end for
6:  $prev \leftarrow PQ.top().second$ 
7: for  $i = 1$  to  $k$  do
8:   while (true) do
9:      $u \leftarrow PQ.pop().second$ 
10:    if  $u == prev$  then
11:       $S \leftarrow S \cup \{u\}$ , break
12:    end if
13:     $ninc_u \leftarrow \sigma(S \cup \{u\}) - \sigma(S)$ 
14:     $PQ.push(ninc_u, u)$ ,  $prev \leftarrow u$ 
15:  end while
16: end for
17: return  $S$ 
```



sub-modularity property, newly computed marginal influence spread increase in line 13 will never grow higher than previous one.

2.6 Independent Path Algorithm

While Kempe et al. and Leskovec et al. try to overcome the NP-hardness of seed selection and reduce the number of evaluating $\sigma(S)$ which is #P-hard, Kim et al.'s algorithm, IPA [10], triggers on reducing the cost of influence evaluation. They use independent influence path as a influence evaluation unit and evaluate the influence $\sigma(S)$ using the combination of influence paths with simple equations. With new influence evaluation method, they used greedy framework to select seed nodes by rewriting several lines of Algorithm 2.5.2.

The core of the IPA algorithm, influence path and its corresponding probability, influence propagation probability which is defined as follows. An influence path from u to v is a acyclic sequence of connected nodes between two nodes u, v which can be represented as $p = \langle v_1 = u, v_2, \dots, v_m = v \rangle (m \geq 2)$. Influence path should be acyclic because a cycle means that influenced node is influenced again which conflicts with the concept of the IC model. The corresponding influence propagation probability of p is defined as

$$ipp(p) = \prod_{i=1}^{m-1} w_{(v_i, v_{i+1})}. \quad (2)$$

To control the number of influence paths, a threshold θ is applied and collected influence paths p only that satisfies $ipp(p) \geq \theta$.

Influence paths starting from a node u are collected by expanding paths through outgoing edges. Paths are expanded until (1) they meets a pre-



visited node, or (2) $ipp(p)$ goes lower than θ . Here comes an example with illustrated figures of expanding paths which was used in [10].

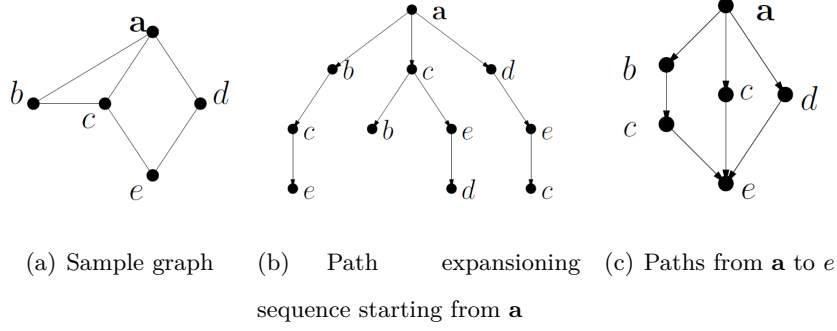


Figure 1: Sample graph and path collection starting from **a**.

Suppose we have an undirected graph, Figure 1(a), which all edge weights are assigned to 0.1 uniformly and threshold $\theta = 0.001$. Then by traversing the graph starting from a , we can expand influence paths from a and obtain Figure 1(b) which represents all influence paths from a . Further, collection of influence paths from a node to another node can be gathered, which is illustrated in Figure 1(c). Those collected influence paths are used to approximate the influence spread $\hat{\sigma}(S)$ which replaces Monte-Carlo simulation based influence spread $\sigma(S)$.

The influence paths are arranged to path collection denoted $P_{A \rightarrow B}$, A and B can be either a node or a set of nodes. More formally, $P_{u \rightarrow v} = \{p | p = \langle u, \dots, v \rangle\}$ and $P_{u \rightarrow T} = \bigcup_{v \in T} P_{u \rightarrow v}$. $P_{S \rightarrow v}$ and $P_{S \rightarrow T}$ are defined in the same manner. Then the influenced area of a node u is defined $O_u = \{v | \langle u, \dots, v \rangle \in P_{u \rightarrow V}\}$ and the influence area of a set $S \subset V$ is defined $O_S = \bigcup_{u \in S} O_u$. Finally,



the approximated influence spread of a node u is

$$\hat{\sigma}(\{u\}) = 1 + \sum_{v \in O_u} \hat{\sigma}_v(\{u\}). \quad (3)$$

The term $\hat{\sigma}_v(\{u\})$ implies the influence that the node v received by u and defined

$$\hat{\sigma}_v(\{u\}) = 1 - \prod_{p \in P_{u \rightarrow v}} (1 - \text{ipp}(p)). \quad (4)$$

which means $1 -$ the probability that none of the influence paths in $P_{u \rightarrow v}$ fails to influence v , intuitively.

Because of the characteristic of the IC model that an active node cannot be activated again, the influence spread of a set of nodes $\hat{\sigma}(S) \neq \sum_{u \in S} \hat{\sigma}(\{u\})$. IPA handles this problem by defining the influence path validity. An influence path $p = \langle u, \dots, v, \dots \rangle$ is valid if and only if $\forall v. v \notin S$. Then, $P_{u \rightarrow v}^{\text{valid}}$, σ_v^{valid} and σ^{valid} are newly defined by using valid influence paths only. Finally the influence path of a set of nodes $S \subset V$ is defined

$$\sigma^{\text{valid}}(S) = |S| + \sum_{u \in O_S} \sigma_u^{\text{valid}}(S) \quad (5)$$

where $\sigma_v^{\text{valid}}(S) = 1 - \prod_{p \in P_{S \rightarrow v}^{\text{valid}}} (1 - \text{ipp}(p))$. By replacing σ in Algorithm 2.5.2 into σ^{valid} , the IPA algorithm is accomplished.

The IPA algorithm has three phases traversing, re-organizing and updating phase. Traversing phase collects influence paths starting from every nodes in V . Re-organizing phase handles the memory allocation of path collections that converts $P_{u \rightarrow V}$ into $P_{V \rightarrow v}$. Finally updating selects new seed node and checks the validity of influence paths.



In the serial version of IPA, each phase has $O(m|O_{avg}|n_{uv})$, $O(m|O_{avg}|n_{uv})$ and $O(k|O_{S \cup \{u\}}|n_{uv})$ time complexity and the entire process has $O(m|O_{avg}|n_{uv}|p_{avg}|)$ space complexity where $|O_{avg}|$, n_{uv} , and $|p_{avg}|$ represent the size of average influenced area, the average number of path collection between two nodes, and average memory consumption per a influence path.

Traversing and updating phases of IPA can be fully parallelized. The time complexity of traversing and updating phase parallel version of IPA is $O((m|O_{avg}|n_{uv})/c)$ and $k(((|O_{S \cup \{u\}}|n_{uv})/c) + (c - 1))$ per each.

2.7 Other Existing Works

Chen et al. evaluated the influence spread of each node by applying simple heuristic using degree information [6]. Chen et al. again introduced fast approximation of influence spread by generating local arborescence [7]. Jung et al. adopted belief propagation before evaluating influence spread of individual nodes [8]. Brogs et al. proposed completely new approach which evaluates the influence spread of all nodes simultaneously with theoretical support [11] and Tang et al. optimized Brogs et al.'s algorithm by automatically tuning the parameter [9].



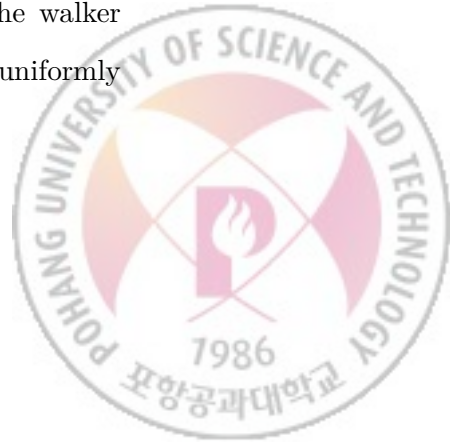
3 Proposed Algorithm

In this section, we propose a novel pruning method Random Walk Pruning (RWP) and a new influence maximization algorithm, RWP-IPA. We show how RWP efficiently selects seed candidates and prune out vast number of uninfluential nodes. Then we apply RWP to the IPA algorithm and propose RWP-IPA which dramatically reduced unnecessary computations in IPA.

3.1 Random Walk Pruning

In the experiment section of [7], authors include the PageRank algorithm [13], a random walk based method to demonstrate the usefulness of their algorithm. The experiment results shows that the running time of the PageRank algorithm is comparable to existing influence maximization algorithms so far. However the influence spread of the PageRank is poor when $k > 3$ because it does not considers influence overlapping between seed nodes. We invent a pruning method based on random walk named Random Walk Pruning (RWP), that conserves the fast execution of the PageRank while do not damaging the influence spread of the final results. RWP successfully prunes out uninfluential nodes and selects seed candidates. We will show an example to protest the effectiveness of RWP using the Stanford dataset.

The main concepts of RWP is quite simple. Starting from a node $u \in V$ a walker randomly walks through the nodes along the edges with reversed direction and walker can walk 1-hop at once. When a walker reaches to a node v from u , we may consider v can influence u because the walker walked along the reversed edge. Suppose that multiple walkers are uniformly



distributed to all nodes in V and start walking through along the reversed graph randomly. After several steps and taking a snapshot of the distribution of random walkers per nodes, we can say that nodes with many walkers have stronger power to influence the others.

To define the random walk more formally to be handled by computer, we performed the random walk in the following way. First, we initialize the random walk score $Score(u) = 1/m$ uniformly for all $u \in V$. In each steps we set the random walk score to 0 for all node and steal the scores to from their out-neighbors $\{v | (u, v) \in E_{out}(u)\}$. The ratio of the score to be stolen is inverse proportion to the in-degree $D_{in}(v)$ of the node. Adopting this ratio, the ratio stealing by the edge (u, v) becomes identical to $w(u, v)$. We evaporate random walk score which is stuck in a node v where $E_{in}(v) = \emptyset$ so to prevent several nodes reigning with the random walk score permanently.

Algorithm 3.1 shows how random walk is performed to score each nodes in the graph. All scores are initialized to zero at the very beginning step (line 1). For all nodes $v \in V$ the random walk scores are gathered from their neighbors through the edges which directions are reversed (line 4-6).

One challenge is that which iteration of random walk score should be used to perform pruning uninfluent nodes. The random walk score keeps oscillating selecting one iteration step does not guarantees the seed nodes to be located in high rank. We propose a new novel pruning method RWP which utilizes the random walk and prune out nodes so that only influential seed candidates remain.

Suppose we have a graph with $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and the optimal



RandomWalk($G, Prev_Score$)

Require: G : social network graph,

$Prev_Score$: random walk score at the previous iteration

Ensure: $Score$: random walk scores

```
1:  $Score \leftarrow [0, \dots]$ 
2: for all  $u \in V$  do
3:    $Sum \leftarrow 0$ 
4:   for all  $(u, v) \in E_{out}(u)$  do
5:      $Sum += Prev\_Score[v] / D_{in}(v)$ 
6:   end for
7:    $Score[u] \leftarrow Sum$ 
8: end for
9: return  $Score$ 
```



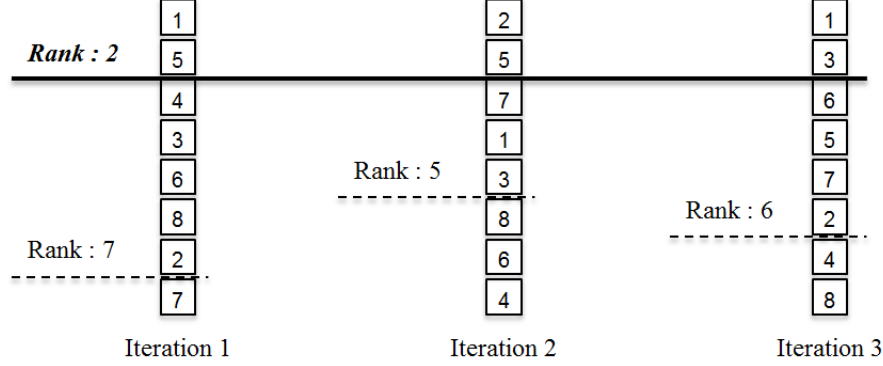


Figure 2: An example of selecting the number of iteration steps and required k' respectively in Random Walk Pruning.

seed nodes set $S = \{1, 2, 3\}$. Figure 2 lustrates an example of the rank of each random walk iteration. In case we use a single iteration to prune out uninfluent nodes, we should remain up to 7, 5 and 6 nodes to ensure that all optimal seed nodes are included in the seed candidates S' (dashed line in Figure 2 and $|S'|$ is 7, 5 and 6 respectively which influence spread should be evaluated per each. However, if we use the combination of the top-2 nodes of three iterations (solid line in Figure 2, we can get $S \subset S' = \{1, 2, 3, 5\}$ and we only have to evaluate the influence spread of 4 nodes which is reduced to half. The definition of seed candidates S' is following:

Definition 4. With given graph $G(V, E, W)$, the number of random walk iterations z and the number of seed candidate nodes in each iteration k' , the seed candidates S' is defined as

$$S' = \cup_{i=1}^z \text{top-}k' \text{ nodes from } \text{RandomWalk}(G). \quad (6)$$



Intuitively we can get the combination of (1-to- z) hop top- k' influential nodes with RWP. Algorithm 3.1 shows the process of RWP below.

RandomWalkPruning(G, k', z)

Require: G : social network graph,

k' : size of seed candidates selected in iteration,

z : number of random walk iterations

Ensure: S' : seed candidates

```

1:  $S' \leftarrow \emptyset$ 
2:  $Score \leftarrow \{1/m, \dots\}$ 
3: for  $i = 1$  to  $z$  do
4:    $Score \leftarrow RandomWalk(G, Score)$ 
5:    $Candidates \leftarrow k'$  nodes with top- $k'$  score
6:    $S' \leftarrow S' \cup Candidates$ 
7: end for
8: return  $S'$ 

```

Even we generate S' from the given graph G , the pruning effect will not be obvious if $|S'| \not\ll m$, moreover computation time may increases for the overhead of processing RPW. Also, the obtained S' should guarantee that the seed nodes included in S' . In the following Lemma 1 and Lemma 2

Lemma 1. *When $k' \ll m$ and $z \ll m$, the seed candidates selected in a single iteration is k' and the expected $|S'|$ is $m(1 - (1 - k'/m)^z)$. In the worst case, $|S'| = zk'$ where there is no overlap between seed candidates selected in each iteration.*



Lemma 2. *If the selected k' seed candidates set in an iteration records y recall of the result of influence maximization algorithm on average, The seed candidates S' obtained by RWP shows $1 - (1 - y)^z$ recall.*

To show the effectiveness of RWP briefly, we show the result of RWP here in Table 2 by using the Stanford dataset which $m = 281K, n = 2.31M$. We obtained S size of 50 using IPA [10] and compared the result with each random walk iterations. We show the minimal k' needed to get all S . Finally we report the minimal k' needed to get all S with different z . The detailed performance of RWP is reported in the Section 4.2.1 later.

Table 2: Effectiveness of RWP in Stanford dataset

Options	minimal k'	$ S' $	ratio of $ S' $ to $ V $
Iteration : 1	84806	84806	30.0834%
Iteration : 2	123353	123353	43.7572%
Iteration : 3	76117	76117	27.0011%
Iteration : 4	133659	133659	47.4131%
RWP ($z = 4$)	105	321	0.1139%

Now we have seed candidates S' which $|S'| \ll m$ In case of the Stanford dataset, we can reduce up to 99.8861% influence evaluations which is #P-hard challenging process. The bigger dataset we use we get the greater influence evaluation processes reduced. RWP can be applied to any algorithm which is challenging to boost #P-hard influence evaluations and using the greedy-approximation framework. When applying RWP to an algorithm, RWP will work as a preprocessing step of the algorithm.

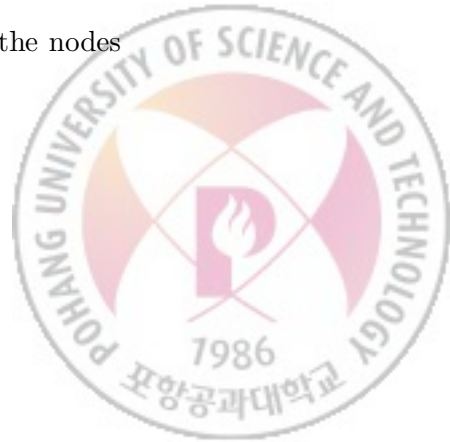


The time complexity of serial version of RWP is $O(z((n + m) + n \log n))$ where n is the number of random walks along to the edges, m is aggregation of random walk scores propagated per each node, and number of random walk iteration z . $n + m$ is needed to run a random walk and $n \log n$ for sorting and finding top- k' scored nodes in each iteration. The space complexity of RWP is $O(2m)$ to preserve the random walk scores of all nodes in current and previous iteration.

Random walk can be easily parallelized by adding few lines of OpenMP meta-programming language expressions. By running the outer for loop in parallel (line 2-8 in Algorithm 3.1) using c threads, the time complexity of random walk reduces to $O((n + m)/c)$ from $O(n + m)$. By applying the parallel random walk, the time complexity of RWP decreases to $O(z((n + m)/c + n \log n))$.

3.2 RWP-IPA

The most time consuming part of the IPA algorithm is the traversing phase which generates all possible influence paths and evaluate the influence spread σ^{valid} (line 2-5 of modified Algorithm 2.5.2). In the DBLP dataset for example, the time consumed in the traversing phase is 51 seconds while the entire process consumed is only 52 seconds. Also, Kim et al. reported that maintaining the priority queue size of $3k$ only does not damage the influence spread [10]. These two phenomenon implies that (1) reducing computation workload in traversing phase is the key of reducing processing time of the IPA algorithm, and (2) generating influence paths for the majority of the nodes



which are not included in top $3k$ influential nodes is waste of time.

Thus, we propose a new influence maximization algorithm RWP-IPA which apply RWP before IPA to initialize the nodes to be traversed and reduce the time consumption in traversing phase. Algorithm 3.2 below shows the procedure how RWP-IPA works.

By revisiting the DBLP dataset example again, we can see that the RWP successfully reduces the time consumed in traversing phase. Table 3 shows the time consumption of IPA and RWP-IPA. Both algorithms use same parameter $k = 50, \theta = 1/320$ and RWP-IPA additionally used $k' = 3k, z = 5$ for RWP.

Table 3: Processing time of IPA and RWP-IPA on DBLP dataset

Algorithm	Traversing(RWP)	Re-organizing	Updating	Total
IPA	51(\cdot)	0	1	52
RWP-IPA	1(3)	0	1	5

The time complexity of serial version of RWP-IPA is $O(z((n + m) + n \log n) + m|O_{avg}|n_{uv} + m|O_{avg}|n_{uv} + k|O_{S \cup \{u\}}|n_{uv})$ and $O((z((m + n)/c + n \log n)) + ((zk'|O_{avg}|n_{uv})/c) + zk'|O_{avg}|n_{uv} + k(((O_{S \cup \{u\}}|n_{uv})/c) + (c - 1)))$ for the parallel version. The space complexity is $O(\max(2m, m|O_{avg}|n_{uv}|p_{avg}|))$ which is the bigger space needed between RWP and IPA because memory need during RWP is freed before IPA.



RWP-IPA (G, k, k', z, θ)

Require: G : social network graph,

k : the number of seed nodes to be selected,

k' : the number of seed candidates in each iteration,

z : number of random walk iterations

Ensure: S : k influential seed nodes

```
1:  $S, PQ \leftarrow \emptyset$ 
2:  $S' \leftarrow \text{RandomWalkPruning}(G, k', z)$ 
3: for all  $v \in S'$  do
4:   evaluate  $\sigma^{\hat{valid}}(\{v\})$ 
5:    $PQ.\text{push}(\sigma^{\hat{valid}}(\{v\}), v)$ 
6: end for
7:  $prev \leftarrow PQ.\text{top}().\text{second}$ 
8: for  $i = 1$  to  $k$  do
9:   while (true) do
10:     $u \leftarrow PQ.\text{pop}().\text{second}$ 
11:    if  $u == prev$  then
12:       $S \leftarrow S \cup \{u\}$ , break
13:    end if
14:     $ninc_u \leftarrow \sigma^{\hat{valid}}(S \cup \{u\}) - \sigma^{\hat{valid}}(S)$ 
15:     $PQ.\text{push}(ninc_u, u)$ ,  $prev \leftarrow u$ 
16:  end while
17: end for
18: return  $S$ 
```



4 Experiments

In this section, we demonstrate the efficiency of RWP with extensive experiments with real world datasets. Also we compare the proposed algorithm RWP-IPA with existing algorithms.

4.1 Experiment Setting

4.1.1 Running Environment

We used a 64-bit Linux machine with Intel Xeon CPU with 6 physical cores (12 hyper-threads) and 24GB memory for experiment. All algorithms were implemented in C++ and used gcc compiler with -O3 optimization.

4.1.2 Datasets

We used five real world datasets which are publicly available. Epinions [17] is a who-trust-whom trust network of customers where nodes are customers and edges are the trust between customers. Stanford [19] is a web graph of `stanford.edu` where nodes represents pages and edges are hyperlinks between them. DBLP [18] is a research collaboration networks of DBLP Computer Science Bibliography available at `dblp.uni-trier.de`. Patents [20] is US patents network of `www.nber.org` from January 1, 1963 to December 30, 1999. LiveJournal [21] is a online social networks where nodes are users and edges are their friendships. All datasets except DBLP are freely available in SNAP [22], and DBLP dataset we used is also available [23] which is preprocessed by Wei Chen. Table 4 shows brief statistics of datasets.



Table 4: Statistics of the datasets

Dataset	Epinions	Stanford	DBLP	Patents	LiveJournal
# of Nodes	75.8K	281K	655K	3.77M	4.85M
# of Edges	509K	2.31M	3.98M	16.5M	69.0M
Average Degree	6.71	8.20	6.10	4.38	14.2
Max In Degree	3032	38606	588	779	13906
Max Out Degree	1798	255	588	770	20293
Direction	Directed	Directed	Undirected	Directed	Directed

Because we have only 24GB memory, we couldn't conduct the experiment with Twitter dataset. RWP-IPA occurs memory failure and TIM⁺ requires 30 GB of memory [9] on Twitter.

4.1.3 Algorithms

We examine the performance of RWP-IPA with multiple existing algorithms. The algorithms are following:

- **Greedy** : the CELF greedy algorithm [5] using 20,000 Monte-Carlo simulation to evaluate the influence spread.
- **SD** : the single discount heuristic algorithm [6] based on the degree of nodes. The degree of a node decreases by 1 when it's neighbor is selected as a seed node.
- **PMIA** : the heuristic algorithm using the local structure [7]. Builds local arborescence structures as local influence region to evaluate influ-



ence spread.

- **TIM⁺** : the state of the art random sampling based algorithm with theoretical support [9].
- **IPA** : the heuristic algorithm which approximates the influence spread by using independent influence path as a influence evaluation unit [10].
- **RWP-IPA** : Our second proposed algorithm which applies RWP and selects seed nodes using IPA.

The source code of TIM⁺, and IPA was provided by their inventors. Because we couldn't get the source code of Greedy and PMIA, we write our own code to run the experiment of both algorithms. To evaluate the influence spread, we used Monte-Carlo simulation with 20,000 iterations.

4.1.4 Parameter Setting

For all existing algorithms, we used the same parameters reported in their paper for all algorithms [5, 7, 9, 10]. For the RWP-IPA, we uniformly set the k' and z parameters uniformly to $k' = 3k, z = 5$ and tuned the parameter θ which is a tradeoff of running time and influence spread. For all dataset, the θ of RWP-IPA was set higher than the θ of IPA, which makes better quality while sacrificing the running time. Table 5 shows the selected parameters for each algorithm.



Table 5: Selected parameters for Greedy, PMIA, IPA, TIM⁺ and RWP-IPA

Algorithm	Epinions	Stanford	DBLP	Patents	LiveJournal
Greedy(R)	$R = 20000$				
PMIA(θ)	1/160	1/160	1/640	1/80	n/a
IPA(θ)	1/320	1/160	1/320	1/40	1/80
TIM ⁺	$\epsilon = 1, l = 1$				
RWP-IPA	$k' = 3k, z = 5$				
(k', z, θ)	1/160	1/160	1/1280	1/640	1/640

4.2 Experiment Results

4.2.1 Effectiveness of RWP

In this section, we run extensive RWP processes on dataset varying parameters. We see how the RWP guarantees the influential nodes selected from IPA by varying the number of iterations z .

First, to select the proper number of the seed set candidates per each iteration k' , we check the seed set of IPA ($k = 50$) covered by the seed candidate of random walk by changing the number of iterations and size of k' . We run the random walk up to 30 and check the ratio of covered seed set with various k' ; total 10000 different k' from 0.01% to 100% of $|V|$. Then we get the average cover of seed set and standard deviation. Figure 3(d) to 3(f) shows the average ratio of covered seed set by differentiating k' in each dataset and Figure 3(a) shows that of entire datasets. Note that the x-axis is log-scaled. The black area is standard deviation of 30 different iteration.



Surprisingly, in most cases, setting k' to less than only 10% of $|V|$ shows 100% cover of seed set produced by the IPA algorithm. In Figure 3(a) we can see that using only $k' = 0.47\%$ of $|V|$ shows average 80% cover of seed set.

Table 6: Required S' to select all the seed nodes of IPA

Dataset	$z = 1$	$z = 2$	$z = 3$	$z = 4$	$z = 5$	RWP(k')
Epinions	87	1174	157	929	383	76(49)
Stanford	84806	123353	76117	133659	5331	311(105)
DBLP	85	237	76	183	138	78(49)
Patents	574627	545	445	1620	22480	177(74)
LiveJournal	2582	1242405	4967	399887	146337	121(59)

Second, to more precisely pick up seed candidates using much smaller k' , we evaluate the required k' and selected $|S'|$ to get all the nodes in the seed set in each dataset. The second to the sixth column in Table 6 shows the selected $S' = k'$ to get all the nodes in the seed set of IPA, and the last column shows the selected $|S'|$ and required k' when using the RWP with $z = 5$. The results in Table 6 shows that RWP effectively selects the seed candidates with using much smaller k' shown in Figure 3(a). Especially, the required k' and selected $|S'|$ in the LiveJournal dataset is 59 and 121 which is 0.0012% and 0.0025% of $|V|$.

Finally we set $z = 5, k' = 3k = 150$ and run RWP to prune out uninfluential nodes and get the seed candidate set S' . The size of the S' for each dataset is shown in the table 7 below.



Table 7: Generated S' after applying RWP with $z = 5$ and $k' = 3k = 150$

Dataset	Epinions	Stanford	DBLP	Patents	LiveJournal
$ S' $	223	418	211	436	304

4.2.2 Processing Time (IPA vs RWP-IPA)

As we briefly discussed in section 3.2, the most time consuming part of the IPA algorithm is the traversing phase which takes more than 75% of entire running time in all datasets. Table 8 shows the detail running time of each phase. Epinions dataset shows that the traversing phase consumes 3.70s and 75.7% of entire dataset which the traversing phase takes the smallest portion among all datasets. In contrast to Epinions dataset, the traversing phase of the DBLP dataset takes 98.2% of the running time which scored the highest portion among 5 datasets.

Table 8: Time consumption of each phase in IPA algorithm

Dataset	Traversing	Re-Organizing	Updating	total
Epinions	3.70	0.28	0.9	4.89
Stanford	8.48	0.11	0.41	9.01
DBLP	31.1	0.2	0.4	31.7
Patents	43.4	0.1	1.5	45.0
LiveJournal	90.6	0.2	1.8	92.6

To get the clear effect of RWP in the view of the running time, we applied the same threshold θ for both the IPA and the RWP-IPA algorithm and get $k = 50$ seed nodes. The θ that we used in this comparison is that of the IPA



in the table 5. We set the parameters z and k' in RWP-IPA as $z = 5$ and $k' = 3k$.

The figure 4 shows the running time difference of IPA and RWP-IPA and time consumed in each phase in both algorithms. In all datasets, we checked that the selected 50 seed nodes were identical because none of the seed node were omitted by RWP, which means that there was no sacrifice in influence spread. Because of the operation process of the RWP-IPA algorithm, the pruning phase emerges and the time consumption of the traversing phase was dramatically reduced while both re-organizing phase and the updating phase consumed the same time as the IPA algorithm. In Epinions dataset, the RWP-IPA algorithm makes 295% speed up which is not a high speed up enough as much as we expected. This phenomenon is caused by the small size of the Epinions graph which means that there was no dramatic difference between the size of original graph and the size of seed candidates. However in case of all the other datasets, the RWP-IPA algorithm made 846%, 3770%, 1916% and 1974% speed up comparing to the IPA algorithm. Especially in the dense graph as DBLP, RWP was highly effective and successfully pruned out uninfluential nodes.

4.2.3 Processing Time (All Algorithms)

By fixing the parameters of each algorithm described in the Table 5, we conducted various experiments in each dataset. The result is illustrated in Figure 5. Note that the y-axis is log-scaled. We omit the result of CELF Greedy on Patent and LiveJournal datasets because the computation time



exceeded 200,000 so that we can't finish the experiment within considerable time. PMIA in the LiveJournal dataset is also omitted because the memory usage of the PMIA algorithm exceed 24GB so that we couldn't conduct the experiment. With all different datasets except Epinions and Stanford, RWP-IPA shows the best speed among the algorithms except SD. However, SD is reported that its influence spread is extremely low [7] so that SD is fast only, but useless. In the Epinion dataset and Stanford dataset, TIM^+ is slightly faster that $k = 50$ is the parameter that TIM^+ can make the fastest speed in Epinions and Stanford, however, the difference between TIM^+ and RWP-IPA is less than 1 seconds. Finally, in the other bigger datasets like DBLP, Patents and LiveJournal, RWP-IPA recorded the fastest running time which is several times faster than the TIM^+ algorithm which is the state-of-the-art.

4.2.4 Influence Spread

With the selected $k = 50$ seed sets returned from the experiments in section 4.2.3, we evaluated the influence spread of selected seed nodes in all datasets. We run the Monte-Carlo simulation with $R = 20000$ times and get the average for influence spread. Figure 4.2.4 shows the influence spread of all algorithms in all 5 datasets. Because all algorithms except Random is approximation of Greedy, algorithms PMIA, IPA, TIM^+ and RWP-IPA shows lower influence spread than that of Greedy. While SD shows the extremely fast seed nodes return in all cases, it turned out that the returned seed nodes of SD has low quality which shows the lowest influence spread among all algorithms except in DBLP dataset. The reason for this is that



DBLP is undirected graph which SD can get a respectable result return. while RWP-IPA shows fast running time in any dataset, it does not sacrifice the quality of the returned seed nodes. In all datasets, because RWP-IPA finished in shorter time than IPA and weaker threshold θ can be applied, RWP-IPA shows better influence spread than that of IPA in all datasets.

4.2.5 Memory Usage

We reported the memory usage of each algorithms. The memory usage is reported in Table 9. The results of greedy and SD are omitted because they use no memory except the graph structure. The texts in **bold** are the algorithm with the minimal memory usage.

Table 9: Memory usage of for PMIA, IPA, TIM⁺ and RWP-IPA

Algorithm	Epinions	Stanford	DBLP	Patents	LiveJournal
PMIA	524MB	2.0GB	4.2GB	14.4GB	N/A
IPA	207MB	597MB	419MB	1.6GB	3.6GB
TIM ⁺	317MB	1.2GB	5.1GB	7.1GB	8.3GB
RWP-IPA	207MB	597MB	419MB	1.7GB	3.8GB

For all datasets, IPA results the lowest memory usage among compared algorithms. However, RWP-IPA shows same memory usage for small datasets, Epinions, Stanford and DBLP, because the space complexity of RWP-IPA is $O(\max(2m, m \cdot |O_{avg}| \cdot n_{uv} \cdot |p_{avg}|))$ which is the bigger one between IPA and RWP. Also, RWP-IPA shows comparable memory usage to IPA in bigger datasets, Patents and LiveJournal.



4.2.6 Parallelization Effect

Finally, in this section, we run the RWP-IPA with various number of cpu cores. The parallel version of RWP-IPA is easily conducted by inserting few OpenMP [12] meta-programming expressions. We measure the speed-up factor by the following equation

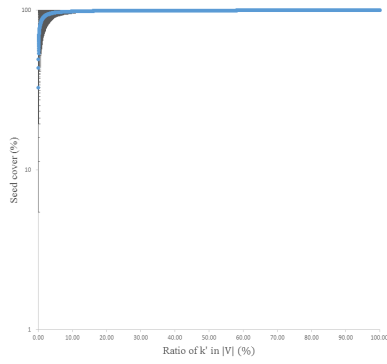
$$\text{speed-up} = \frac{\text{processing time of serial algorithm}}{\text{processing time of parallel algorithm}}. \quad (7)$$

In the ideal case, for c cores, c speed-up is expected. However, in general, the speed-up factor records lower than c because (1) certain part of program cannot be serialized, like synchronization section, parts that cannot be serialized, (2) overhead of managing threads and swapping threads, (3) shared memory access, allocation and de-allocation.

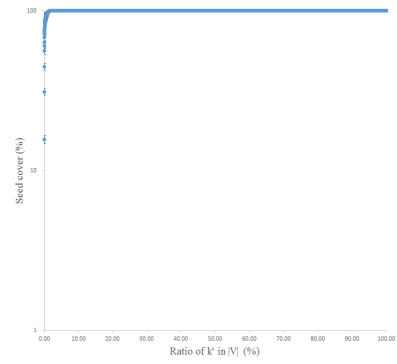
Figure 7 shows the speed-up factor growth of the parallel RWP-IPA. The diminishing increasement is commonly visible for all dataset. The is phenomenon is explainable by Amdahl's law [24] which speed-up factor never grows linearly as the cpu cores increases.

For the small and sparse dataset like Epinions and Stanford, the parallelization effect is not significant. However, small but dense dataset like DBLP shows visible speed-up which takes long time in RWP and traversing influence paths. Large graph like Patents and LiveJournal datasets shows a stable speed-up because the parallelizable part consumes much more processing time than synchronization part.

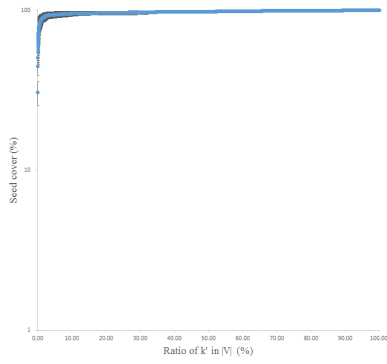




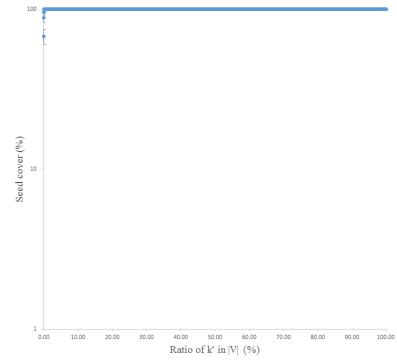
(a) Total graphs



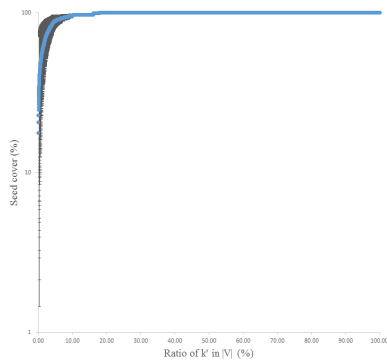
(b) Epinions



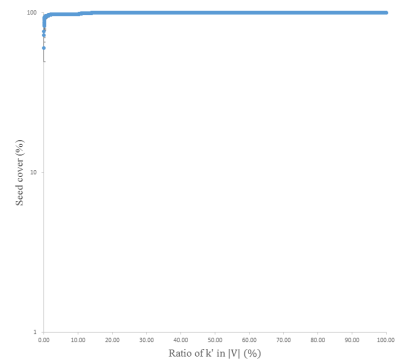
(c) Stanford



(d) DBLP



(e) Patents



(f) LiveJournal

Figure 3: Average coverage of seed nodes of IPA varying the size of k' in each iteration.



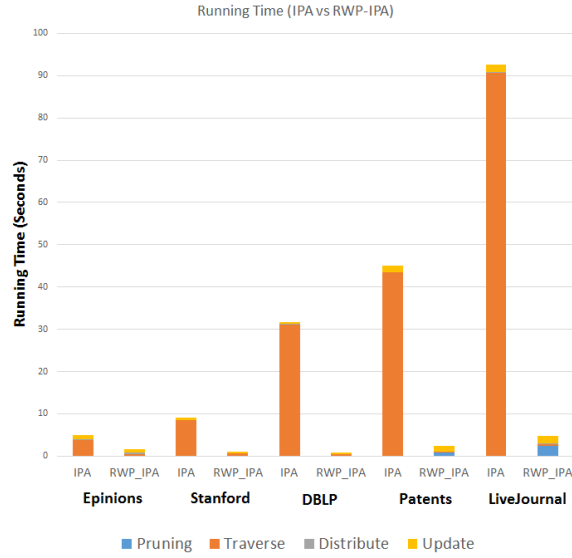


Figure 4: Running time comparison of IPA and RWP-IPA

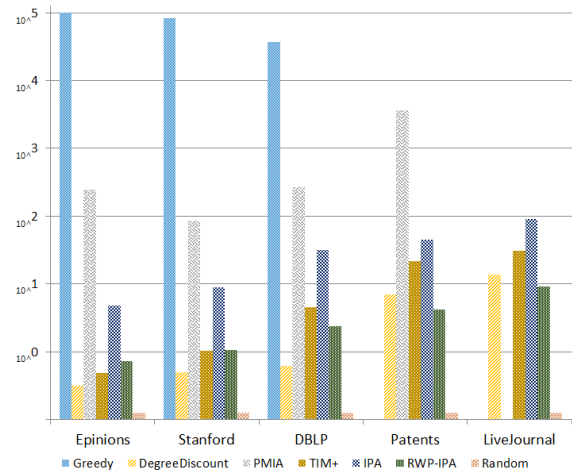
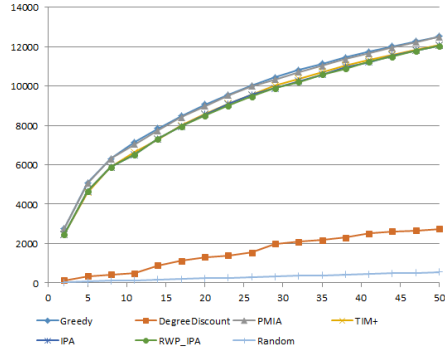
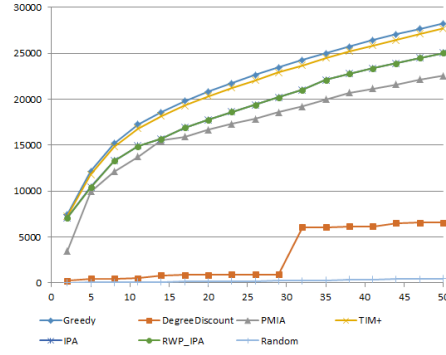


Figure 5: Processing time of each algorithm in five datasets.

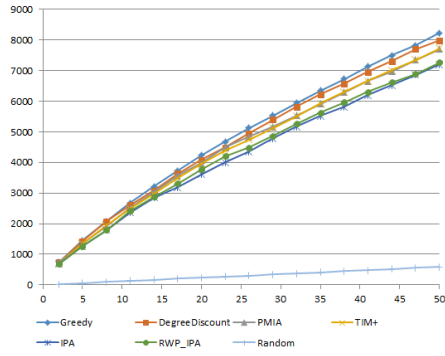




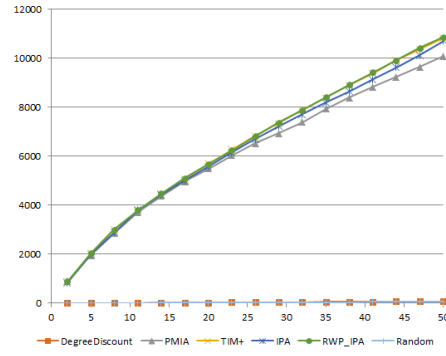
(a) Epinions



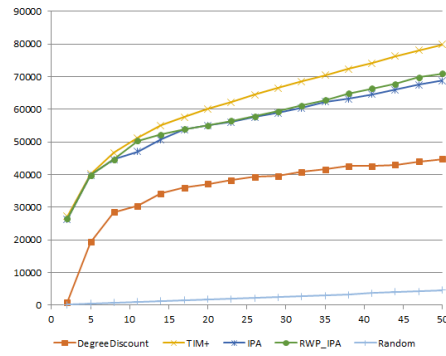
(b) Stanford



(c) DBLP



(d) Patents



(e) LiveJournal

Figure 6: Average coverage of seed nodes of IPA varying the size of k' in each iteration.



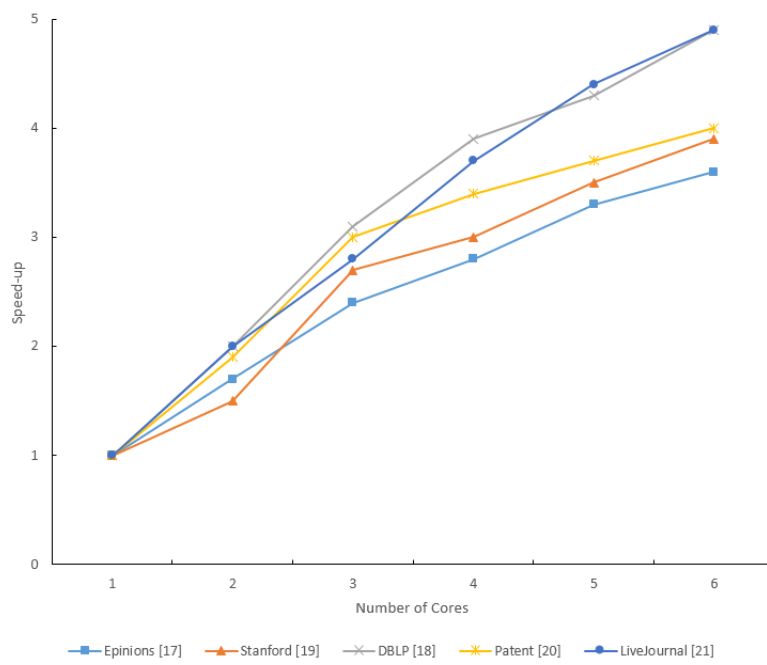


Figure 7: Speed-up of parallel RWP-IPA.



5 Conclusion

To eliminate unnecessary influence evaluations and increase the total speed of influence maximization, we proposed RWP in IC model that prunes out uninfluential nodes effectively. The main idea of RWP is that the combination of random walk iteration which runs only in $O(z(m+n)/c)$ contains the most influential seed nodes and the remainder nodes are ignorable. Then we extend the IPA by applying RWP and proposed the RWP-IPA algorithm to reduce the total number of influence spread evaluation which is #P-hard problem. The experiment results shows that our algorithm is outstandingly fast and uses less memory than other state-of-the-art algorithms.



References

- [1] H. Ma, H. Yang, M. R. Lyu, and I. King, "Mining social networks using heat diffusion process for maketing candidates selection", *Proceedings of the 17th ACM conference on Information and knowledge management*, ser. CIKM'08. New York, NY, USA: ACM, 2008, pages 233-242.
- [2] P. Domingos and M. Richardson, "Mining the network value of customers", in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'01. New York, NY, USA: ACM, 2001, pages 57-66
- [3] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral maketing", in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'02. New York, NY, USA, 2002, pages 61-70
- [4] D. Kempe, J. Kleingerg and E. Tardos, "Maximizing the spread of influence through a social network", in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'03. New York, NY, USA: ACM, 2003, pages 137-146
- [5] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen and N. Glance, "Cost-effective outbreak detection in networks", in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'07. New York, NY, USA: ACM, 2007, pages 420-429



- [6] W. Chen, Y. Wang and S. Yang, "Efficient influence maximization in social networks", in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'09. New York, NY, UAS: ACM, 2009, pages 199-208
- [7] W. Chen, C. Wang and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks", in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'10. New York, NY, USA: ACM, 2010, pages 1029-1038
- [8] K. Jung, W. Heo and W. Chen, "IRIE: Scalable and Robust Influence Maximization in Social Networks", in *Proceedings of the 2012 IEEE International Conference on Data Mining*, ser. ICDM'12. Washington, DC, USA: IEEE Computer Society, 2012, pages 918-923.
- [9] Y. Tang, X. Xiao and Y. Shi, "Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency", in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, ser. SIGMOD'14. New York, NY, USA: ACM, 2014, pages 75-86.
- [10] J. Kim, S. Kim and H. Yu, "Scalable and Parallelizable Processing of Influence Maximization for Large-Scale Social Networks", in *Proceedings of the 2013 IEEE International Conference on Data Engineering*, ser. ICDE'13. 2013, pages 266-277



- [11] C. Brogs, M. Brautbar, J. T. Chayes and B. Lucier, "Maximizing social influence in nearly optimal time", in *ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA'14. 2014, pages 946-957.
- [12] L. Dagum and R. Menon, "Openmp: An industry-standard api for shared-memory programming", in *IEEE Comput. Sci. Eng.*, vol. 5, pages 46-55, Jan, 1999.
- [13] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine", in *Seventh International World-Wide Web Conference*, ser. WWW'98. Amsterdam, The Netherlands: Elsevier Science Publishers, 1998, pages 107-117.
- [14] K. Saito, R. Nakano and M. Kimura, "Prediction of information diffusion probabilities for independent cascade model", in *Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part III*, ser. KES ' 08. Berlin, Heidelberg Germany: Springer-Verlag, 2008, pages 67-75.
- [15] A. Goyal, F. Bonchi and L. V. Lakshmanan, "Learning influence probabilities in social networks", in *Proceedings of the third ACM international conference on Web search and data mining*, ser. WSDM'10. New York, NY, USA: ACM, 2010, pages 241–250.
- [16] A. Goyal, F. Bonchi and L. V. S. Lakshmanan, "A data-based approach to social influence maximization", in *Proceeding of the VLDB Endowment*, ser. PVLDB'11., vol. 5, no. 1, pages 73–84, Sep. 2011.



- [17] M. Richardson, R. Agrawal and P. Domingos, "Trust Management for the Semantic Web", in *The Semantic Web - ISWC 2003*, ser. Lecture Notes in Computer Science, D. Fensel, K. Sycara, and J. Mylopoulos, Eds. Springer Berlin / Heidelberg, 2003, vol. 2870, pages 351–368.
- [18] M. Ley, "DBLP: Some Lessons Learned", in *Proceeding of the VLDB Endowment*, ser. PVLDB'09., vol. 2, no. 2, pages 1493-1500, Aug. 2009.
- [19] J. Leskovec, K. Lang, A. Dasgupta and M. Mahoney, "Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters" in *Internet Math.*, vol. 6, no. 1, pages 29-123, 2009
- [20] J. Leskovec, J. Kleinberg and C. Faloutsos, "Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations", in *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'05. New York, NY, USA: ACM, 2005, pages 177-187.
- [21] L. Backstrom, D. Huttenlocher, J. Kleinberg and X. Lan, "Group Formation in Large Social Networks: Membership, Growth, and Evolution", in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD'06. New York, NY, USA: ACM, 2005, pages 44-54.
- [22] J. Leskovec, <http://snap.stanford.edu/papers.html>.



- [23] W. Chen, <http://research.microsoft.com/en-us/people/weic/projects.aspx>.
- [24] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities", in *Proceedings of the April 18-20, 1967, spring joint computer conference*, ser. AFIPS'67 (Spring). New York, NY, USA: ACM, 1967, pages 483-485.



감 사 의 글

2006년 아무것도 모르는 채로 포항에 온 지 벌써 9년이 지났습니다. 무학과 이후 둘째 해부터 전과를 하느라 방황하고 5년만에 학부를 졸업하고, 통합과정에서 석사과정으로 전환하느라 또다시 2년을 허비했습니다. 그래도 아직 늦었다고 생각하지 않고 일을 추진한 결과 이렇게 졸업할 수 있게 되었습니다. 여태까지 저를 믿고 제 선택을 존중해주시며 저를 응원해 주신 여러분들께 감사의 말을 전합니다.

우선 학부생 시절부터 저를 지켜봐 주시고 대학원에서도 지도해 주셨던 유환조 교수님께 감사드립니다. 교수님의 덕분에 데이터를 다루는 즐거움을 알 수 있었습니다. 통합과정에서의 전환을 허락해 주시며 저의 선택을 존중해 주신 점 감사히 생각합니다. 그리고 다양한 수업을 통하여 제게 데이터베이스의 지식을 심어주신 황승원 교수님, 미국에 계심에도 수많은 디스커션으로 연구에 도움을 주신 황정현 교수님께 감사의 말을 전합니다.

무려 5년이란 긴 시간동안 같이 동거동락해온 연구실의 형누나동기동생들, 정말 이런 예측불가의 저를 인내심 게이지를 업그레이드 해가며 지켜봐 주신 모두들 너무 고맙네요. 저보다 긴 시간을 포항에서 지내신 진하형... 드디어 포항을 뜨시고 미국 진출! 제가 속도 엄청 썩였을텐데 받아주셔서 감사합니다. 형과의 연구 경험은 제 성장에 있어 한 열 획 정도를 그었다고 해도 모자랄 겁니다. 저는 이제 형의 뒤를 이어 형의 추억 속의 그 곳으로..... 그닥 긴 시간을 같이 한 것은 아니었지만 묵직한 풍림화산과도 같은 모습을 보여주신 영대형, 중국에 있을 때 컨퍼런스콜이 공짜라 한번 전화를 몇 번 걸었는데 오랜만인데도 반갑게 대해 주셔서 놀랐고 감사했습니다. 어여 박사과정도 끝내시고 한국으로 돌아..오시나요? 그리고 졸업을 하고도 아직 포항에 계신 DM 연구실의 정신



적이고 실질적인 scv 진오형, 형이 없었으면 지금의 연구실은 존재하지 않았을 거라고 생각합니다. 그동안 감사했고 국수 얻어먹을 날을 학수고대 하겠습니다. 그리고 자주 놀러갔던 계곡에서의 추억은 잊지 못할겁니다. 하루빨리 포항을 탈출 하시고 광명을 찾으세요. 신화 속의 미노타우르스와도 같이 연구실의 우직함을 담당하시는 성철이형, 꺾을 수 없는 의지로 연구를 이어 나가시더니 어느샌가 프로포절도 하시고 드디어 포항 탈출이 눈앞이시네요. 형의 우직한 서포트는 잊지 않을겁니다. 여러모로 배려심 많으시고 눈치 없는 제가 봐도 저 때문에 뒷목 잡고 딴박 하신 적이 한두번이 아니셨을텐데 꺾꺾 눌러담으시느라 수고하셨습니다. 앞으로는 먼저 취직해서 소고기 많이 사드릴게요! 소고기 하면 또 빼 놓을 수 없는 삼성맨 채용이형. 뽕때마다 맨날 얻어먹기만 해서 너무 고맙고 죄송하네요. 식사 졸업을 한다고 했을 때 이것저것 많이 상담해주시고 고민해 주신거 감사합니다. 그리고 예전에 연구실 구조가 오픈형일 때 형 덕분에 제가 카라 덕후가 됐습니다.... 아 지금은 레인보우로 갈아탐. 요즘 거의 저를 헤드헌터 급으로 돌봐주시는 일환이형, 형 덕분에 제 연봉이 1.5배는 뽕튀기 되었네요. 곧 결혼하시는거 축하드리고 신혼집에 외양간 한칸 두시길 바랍니다. 형에게는 소고기 사주는 정도로 안됨. 송아지 한마리 놓아드리죠. 이제 만약 제가 그 곳으로 입사하게 되면 쪽 같이 일하겠네요. 잘 부탁 드리고 앞으로의 질문러쉬 대비하세요. 아 그리고 소개팅 알아봐 주시는건 고마운데 강제집행만은 좀... 연구실의 잡다한 소품과 건강을 책임지시는 태훈이형. 서울로 가시더니 솔로를 탈출하시던데... 형만 그러기임요? 저도 이제 포항 뜨면 될.. 까요? 될놈될, 안될안. 앞으로 연구주제 맘에드는거 찾으시고 후딱 프로포절 디펜스 마치고 박사님 되시길 바라요. 요즘 휴가만 받으시면 스쿠버다이빙을 다니시느라 바쁘신 선이 누나, 덕분에 바다사진 자주 구경합니다. 그런데.. 이 추운때에 다이빙하면... 춥지...않으세요? 아 거긴 따뜻한 남쪽나라라 다른가?



서울에서 자주 뵙고 그 제 2의 그분에게도 안부 전해주세요. 요즘 면접 다니면서 자주 뵙는 영철이형, 이젠 뭐 당분간은 엄청 자주 뵙겠네요. 형 기다려요 제가 형 소개팅 곧 물어다줄 ㅇㅇ 기달. 이러다 언젠가 한대 맞을 것 같지만 그만 두진 않을테야. 후배인듯 후배가 아닌듯 후배 같은 너.....가 아니라 형.... 동은이형 그동안 제가 야식 먹자 할때마다 참으시느라고 고생 많으셨습니다. 앞으로 탄수화물이랑 사료좀 줄이시고 고기를 드세요. 인간은 육식동물입니다. 그래야 살이 찌죠. 누구보다도 각별한 우리의 동기 전석사님. 여기저기 회사들을 알아볼 때 너한테 들은 조언들 참고도 많이 했고 진짜 도움이 됐다. 비록 너가 박사과정으로 돌아올 때 연구실에서 맞이해 주진 못하겠지만 형은 너의 귀환을 믿어 의심치 않는단다. 그리고 폭사님에게 안부좀 전해줘. 우리 연구실 최고의 약쟁이 뽕규동, 요즘 서울서 고분분투 한다고 들었는데 힘내고 건승하길 바란다. 언제 함 보고 술 한잔 하면서 너의 서울에서의 썰을 들어보도록 하자. 나를 한방에 중국인으로 만든 그 사진은 영원히 잊지 않으나. 후배인듯 후배가 아닌듯 후배 같은 너 2... 이홍창 너임마 포스빌에서 5 시에 보기로 했으면 5 시에 좀 나와 임마 5 시 5 분에 전화해서 나오게 하지 말고. 너도 이번에 같이 졸업하던데 너 성격이면 어딜 가서도 잘 적응하고 앞가림 잘 할 테니 걱정하지 않는다. 너 근데 구두를 어떻게 신은거냐? 땃국물 천지다 신발 닦게 1000 원 내놔. 아무리 두 학번이 차이가 나지만 동갑인데도 형형 하면서 대해준 고마운 우리 연구실의 (구) 귀여움 담당 동현이(여기서 모든 꼬임이 시작되었지). 이제는 병주가 들어왔으니 귀염둥이 타이틀은 물려주게 되었지만 이젠 연구실의 기둥이 되었구나. 내가 나가게 되어 이것저것 부담이 늘었을텐데 미안하고 너라면 연구실의 핵심으로서 후배들의 본보기가 될 수 있을거라고 믿는다. 지금 준비하는 KDD 잘 되길 진짜로 바라고. 근데 7살은 진짜.... 너 전생에 뭐했냐? 연구실에 들어온지 첫 해 만에 다양한 멘붕을 겪고 있는 찬영이. 같이 경희대에 끌려다니느라 너가 고생이 많다. 어서



와, 포항은 처음이지? 많이 심심해 하는 것 같은데 적응 잘 해라. 몇년 까짓거 금방이여. 정의의 이름으로 앞으로 연구실의 돌격대장 노릇을 해줄 수 있길 바란다. 나만의 파티션인줄 알았는데 어느새 빈 자리를 꿰차고 들어온 (작은) 진하. 형이 너 흑마늘 다 가지라 할 때 가져가지 그랬니... 요즘 200 명짜리 채점 하느라 아주 죽을 맛이던데 앞으로 어사인에 랜덤 함수는 넣지 말도록. 자동채점 프로그램을 만들 수 없어. 멘붕 빨리 극복하고 너 할 일도 해야지, 파이팅. 예진아 난 너가 정말 부러워... 그 우월한 기력스..... 아니야 말하니까 슬퍼진다. 흑마늘 맛있게 먹고 우리 연구실 유일의 바이오 라인을 멋지게 살려서 유니크한 박사가 되었으면 좋겠다. 그리고 술 들어갔다고 반말은 그렇다 쳐도 때리는건 좀 참아주고... 언제나 “오→빠→↓” 라고 부를때마다 무언가 긴장을 하게 만들던 유진이. 우리 연구실의 차기 에이스로서 폭풍성장 하는 모습이 정말 멋지다. 못미더운 프로젝트 책임자랑 같이 KT 프로젝트를 하느라 수고했고 내가 앞으로도 안 혼나도록 잘 할게. 마지막으로 연구실의 신흥 귀요미로 동현이를 밀어낸 과묵하지만 가끔씩 기분 좋으면 표정으로 말하는 병주. 형이랑 피카츄 사길 잘했지? 학부도 포항 대학원도 포항이라 한참 포항에 살게 되겠지만 곳곳하게 이겨내길. 전부 다 쓰고 보니 제가 정말 많은 동료들과 지내왔네요. 여러분과 함께 한 대학원 생활을 축복으로 여기고 정말 여러분을 만나 다행이고 감사히 여깁니다.

지금은 다들 여기저기로 진출했지만 아직도 떠들면 즐거움과 편안함을 선사해주는 우리 11분반의 선배 동기 후배들, 무턱대고 마술을 하고 싶어서 만들었던 동아리에 들어와서 따라와 주었던 많은 동기와 후배들, 성비 1:0 (Inf:0)의 기적적인 비율을 자랑하는 우리 컴공과 06학번 친구들과 나와의 실질적인 학부생활을 같이 한 컴공과 07학번 동생들, 북경에서의 생활을 외롭지 않게 해준 MSRA의 동료들, 그리고 중국에서 막 돌아와서 적응기간이 필요하던 저를 기쁘게 맞아준,



죽이 잘 맞아서 같이 잘 놀고 방에서도 이것저것 마셨던 마지막 룸메이트 희탁이.
여러분들이 있었기에 제가 졸업을 할 수 있었습니다. 앞으로 어디에 가더라도
여러분과 좋은 인연을 이어가길 기원합니다.

마지막으로 하나밖에 없는 외동아들의 9년간의 타지생활을 묵묵히 응원해
주신 부모님께 큰 절을 올립니다. 멀다는 핑계로 잘 올라가지도 않고 바쁘다는
핑계로 전화도 자주 못 드렸지만 이제부터 집안에도 도움이 되는 기둥이 되겠습
니다.

그 외에도 이 곳에 적지 못한 많은 분들께 감사의 말을 전합니다. 웃기만
스쳐도 인연이라고 언제나 웃는 얼굴로 여러분과 함께하고 싶습니다. 즐거움을
공유하는 친구로, 서로를 자극하는 선의의 경쟁자로, 힘든 나날을 위로해주는
든든한 동반자로서 서로를 위해줄 수 있었기에 저, 여러분, 우리가 성장할 수
있었다고 생각합니다. 저 역시 누군가에게 도움을 주고 누군가의 든든한 베타자
누군가의 힘이 되었을 것이라 회고하고 앞으로 맞이하게 될 새로운 세계에 기대감
반 두려움 반의 한 발을 내딛으며 이 글을 마칩니다.

2014년 12월 26일.

김승걸 드림



Curriculum Vitae

Name : Seungkeol Kim (김 승 결)

Education

2006.3 - 2011.2 : 포항공과대학교 컴퓨터공학과 (B.S.)

2011.3 - 2015.2 : 포항공과대학교 컴퓨터공학과 (M.S.)

