Source Detection via Random Walks

Kian Shamsaie, Saber Salehkaleybar, Patrick Thiran

October 1, 2023

1 Main Body

1.1 Problem Formulation

Consider a graph $G = (V, E)$ with the vertex[1] set V and the edge set E. We denote the neighbors of a vertex $v \in V$ by $N(v) = \{v' | (v, v') \in E\}$. Moreover, for any subset of vertices such as $V' \subseteq V$, we define the neighborhood of V' (denoted by $\mathcal{N}(V')$) by considering all the vertices that are adjacent with vertices in V', i.e., $\mathcal{N}(V') = \{v | v \in V \backslash V', \exists v' \in V', v \in N(v')\}$.

We consider the susceptible-infected (SI) model for the propagation of infection over this graph as follows. At the time 0, a source node (aka *patient zero*) denoted by s becomes infected. At any time step t, we show the subset of nodes that are infected by I_t. By definition, $I_0 = \{s\}$ at time 0. The set of infected nodes at time $t+1$ is determined as follows. A subset of vertices in $\mathcal{N}(I_t)$ which were not infected before, get infected. We define t_v as the time of infection of vertex v.

Our goal is to develop an algorithm to find the source of the infection s with high success probability. Success probability $P(Success)$ is defined as the probability of finding the exact source by using the algorithm. We can use two types of queries: contact query and test query. By contact query for a node v, all neighbors of v are provided. By test query for a node v, the time of infection t_v is provided with some probability $1 - p$ where $0 < p < 1$. There are several reasons in real-world scenarios where the exact infection time is unknown. For example, in case of disease propagation, a patient might be asymptomatic or fail to receive a test that determines the infection, or in other cases, the accurate data from patients might not be accessible due to security reasons or due to loss of information records. Therefore, we consider a probability p that models all these uncertainties about whether the time of infection for any patient is known. We define the total number of contact queries and test queries by n_{cont} and n_{test}. The costs of each contact query and test query are c_{cont} and c_{test}, respectively. We assume the cost of a test query is much greater than the one for a contact query, i.e., $c_{cont} \ll c_{test}$., because contacts are usually known and easier to find but testing a patient is usually more expensive and might require particular clinical actions. The total cost function that we would like to minimize can be formulated as a weighted sum $c_{cont} n_{cont} + c_{test} n_{test}$. The constant numbers c_{cont}, c_{test} can vary and is dependent on the problem and circumstances. Depending on how we find contact and which technology we use to test patients, the cost of these two queries, and therefore these two constants might change. Therefore, we need to develop an algorithm that is flexible in terms of test-contact query trade-off and can be tuned to perform well in any situation with any arbitrary costs c_{cont}, c_{test} where we have a specific budget for each query type. We will propose an algorithm that satisfies this flexibility and we will also provide some theoretical results on the probability of finding the source and we will give some upper bounds for n_{cont} and n_{test}. For theoretical results, we assume that the underlying graph G is a tree.

[1] In the paper, we use the terms "node" and "vertex" interchangeably.

1.2 Proposed Method

We propose the Random Walk Search (RWS) algorithm for the problem of locating the source of an infection which can significantly improve the success probability in finding the source and it has a controllable test-contact query trade-off. We present the pseudo-code of RWS in Algorithm 1:

Algorithm 1 RWS algorithm

```
s' ← input node
S ← [ ]
Q ← [ ]
while S[−1] ≠ s' do
    S.append(s')
    for u ∈ N(s') do
        RW ← a nonintersecting random with length f(t_{s'}) starting with edge (s', u)
        if RW[−1] has not been tested then
            Q.append(RW[−1])
        end if
    end for
    for v ∈ Q do
        test v
        if t_v is known and t_v < t_{s'} then
            s' ← v
        end if
    end for
end while
return S[-1]
```

We start from a node s' that has been infected given in the input of the algorithm. At each step, we update s' in the following manner: For any neighbor node of s' such as $u \in N(s')$, we generate a non-intersecting random walk of length $f(t_{s'})$ starting from s' and containing the edge (s', u), where $f : \mathbb{R} \to \mathbb{N}$. The last node on each random walk is then tested and if $t_v < t_{s'}$ for some node v, we update s' to v. We can also change the pseudo-code above to derive new versions of RWS: If such v with $t_v < t_{s'}$ is not found, we then repeat the aforementioned procedure by generating a new set of random walks again or test other nodes on the traversed random walks to find such i. The algorithm terminates when $t_i = 0$ is found or no improvements can be made after a certain number of steps of the algorithm.

The choice of the function f can control the test-contact query trade-off. Running random walks with small lengths requires a large number of test queries and a small number of contact queries We consider several versions of the algorithm according to the choice of f. For example, we can choose a constant function $f(t) = c$, or time variant forms such as $f(t) = t$ and $f(t) = \frac{t}{2^k}$ for $k = 0, 1, 2, \cdots$.

1.3 Theoretical Results

We assume that the underlying graph G is a tree. We define the transmission tree as the rooted directed tree which represents an edge from a to b for each transmission of disease from a to b where b is infected for the first time. Assume that v_0 is the source and we start our search from a node v_l at distance l from v_0. The transmission path from v_0 to v_l is $(v_0, v_1, \cdots, v_l)$.

1.3.1 RWS with $f(t) = c$

In this version, the length of each random walk is always c therefore it is impossible to guarantee finding the exact source. the best possible guarantee is reaching to $d(s', s) < \frac{c}{2}$ for some s'. We define the $p_c(l)$ as the probability of reaching to $d(s', s) < c$ for some s' starting from l.

Theorem 1. if each node in the tree G has degree at most $d + 1$, then $p_c(l) \geq (\frac{1-p}{d^{c-1}})^{\lfloor \frac{l}{c} \rfloor}$

Proof. The probability of v_{l-c} being the last node of the random walk passing through v_{l-1} is at least $\frac{1}{d^{c-1}}$ therefore $p_c(l) \geq (\frac{1}{d^{c-1}})(1-p)p_c(l-c)$ which proves the theorem.

1.3.2 RWS with $f(t) = t$

We take the function $f(t) = t$ and we have some assumptions: the transmission tree is a d-ary tree and we use binary search to find the minimum infection time on each random walk.

We define p_i as the probability of finding s when we start our algorithm from v_i.

Theorem 2. $p_l \geq (1-p)(1-p(1-\frac{1}{d}))^{l-1}$ for each $l \geq 1$.

Proof. The probability that a random walk contains v_0 is $\frac{1}{d^{k-1}}$. For each $i \in \{1, 2, \ldots, k-1\}$, the probability that a random walk contains v_{k-i} but no closer node on the transmission path is $\frac{1}{d^{i-1}} - \frac{1}{d^i}$. Therefore, we can write the following recursive relations for $k \in \{1, 2, \ldots, l\}$:

$$p_0 = 1$$

$$p_k = (1-p)(\sum_{i=1}^{k-1} p_{k-i}(\frac{1}{d^{i-1}} - \frac{1}{d^i}) + \frac{1}{d^{k-1}})$$

By simplifying the above recursion we have:

$$p_k = p_{k-1}(1 - p(1 - \frac{1}{d}))$$

By solving the recursion we have:

$$p_l = (1-p)(1-p(1-\frac{1}{d}))^{l-1}$$

We can calculate the final success probability by summing over nodes h as the starting node of our algorithm:

$$P(Success) = P(d(s,h) = 0) + \sum_{l=1}^{\infty}(1-p)(1-p(1-\frac{1}{d}))^{l-1}P(d(s,h) = l)$$

This is a significant improvement for small d compared to other methods. In Greg's paper parameter $p = \frac{p_a}{p_a + (1-p_a)(1-p_h)}$ is used and the final success probability is:

$$P(Success) = \sum_{l=0}^{\infty}(1-p)^l P(d(s,h) = l)$$

Now we give upper bounds on the number of test queries. Let q_i be the expected number of test queries when we start our algorithm from v_i. We assume that t_v is known for all nodes v on the transmission path.

Theorem 3. $q_l \leq (d+1)l \log t_l$

Proof. Starting from a node v_k, first, we need to test at most $\log t_k$ nodes on each of $d+1$ random walks originating from v_k, which takes $(d+1)\log t_k$ test queries. Then for each $i \in \{1, 2, \ldots, k-1\}$, the probability that a random walk contains v_{k-i} but no closer node on the transmission path is $\frac{1}{d^{i-1}} - \frac{1}{d^i}$. Therefore, we can write the following recursive relations for $k \in \{1, 2, \ldots, l\}$:

$$q_0 = 0$$

$$q_k = \sum_{i=1}^{k-1} q_{k-i}(\frac{1}{d^{i-1}} - \frac{1}{d^i}) + (d+1)\log t_k$$

By simplifying the above recursion we have:

$$q_k \leq q_{k-1} + (d+1)\log t_k$$

By solving the recursion we have:

$$q_l \leq (d+1)l\log t_l$$

Now we give upper bounds on the number of contact queries. Let r_i be the expected number of contact queries when we start our algorithm from v_i.

Theorem 4. $r_l \leq (d+1)lt_l$

Proof. Starting from a node v_k, generating each of the $d+1$ random walks with length t_k originating from v_k, takes at most t_k contact queries, which takes $(d+1)t_k$ contact queries in total. Then for each $i \in \{1, 2, \ldots, k-1\}$, the probability that a random walk contains v_{k-i} but no closer node on the transmission path is $\frac{1}{d^{i-1}} - \frac{1}{d^i}$. Therefore, we can write the following recursive relations for $k \in \{1, 2, \ldots, l\}$: We can write the following recursive relations:

$$r_0 = 0$$

$$r_k \leq \sum_{i=1}^{k-1} r_{k-i}(\frac{1}{d^{i-1}} - \frac{1}{d^i}) + (d+1)t_k$$

By simplifying the above recursion we have:

$$r_k \leq r_{k-1} + (d+1)t_k$$

By solving the recursion we have:

$$r_l \leq (d+1)lt_l$$

1.4 Experiments

We have used the susceptible-infected (SI) model in all of our experiments. We defined a constant *pi* parameter as the probability of transmitting disease over an edge from an infected node to a susceptible node at each time step. We set a limit for the number of tests on each time step. We consider four metrics for evaluating the algorithms: (1) number of contact queries, (2) number of test queries, (3) success probability (aka accuracy) that the exact source is found, (4) distance of the actual source to the final node found in the algorithm. All these metrics are averaged across 100 random runs of the algorithm. On each run, a random node in G is selected as the initial input node for the algorithm.

Experiments with constant $f(t)$ are shown in Figure 1 and Figure 2. We can see by increasing the length of random walks, the number of tests will decrease. Experiments on d-ary trees are shown

in Figure 3 and Figure 4. We also compared our method with the LS algorithm in Figure 5 and Figure 6 on two different networks.

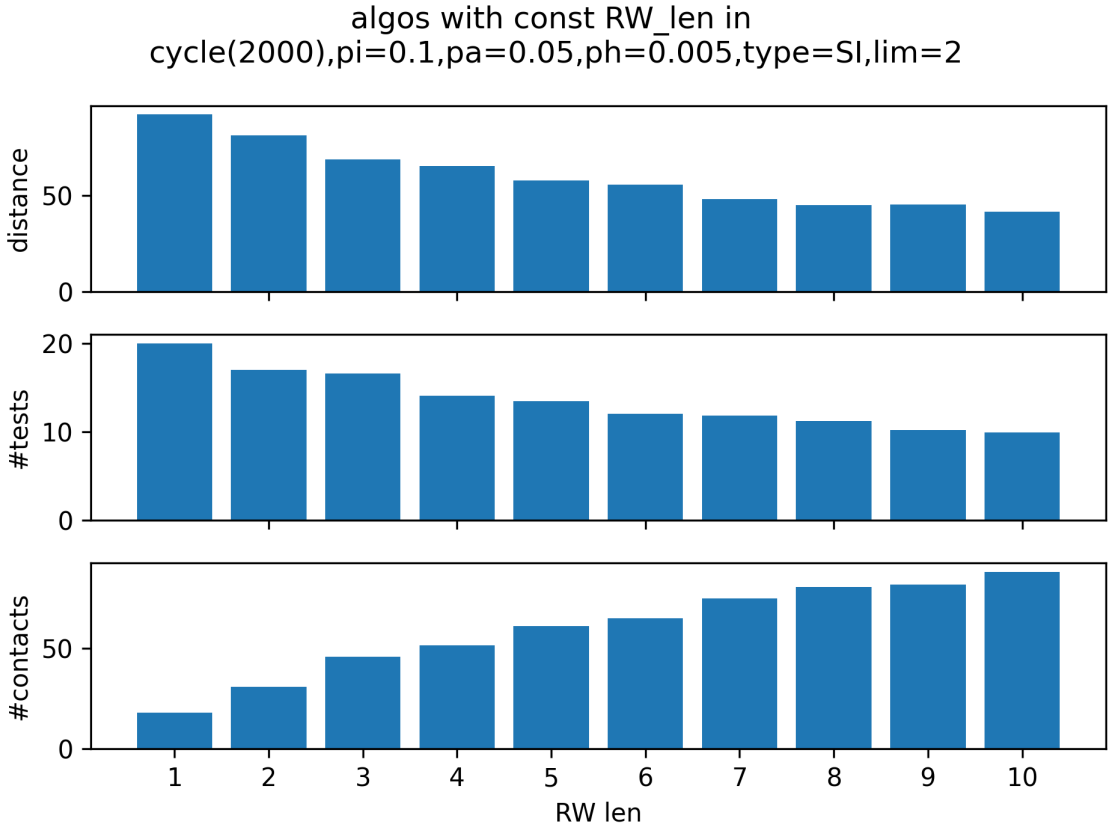

Figure 1: G is a cycle with 2000 nodes. By increasing the length of the random walks, the number of tests and distance to the source has decreased, but the number of contact queries has increased.

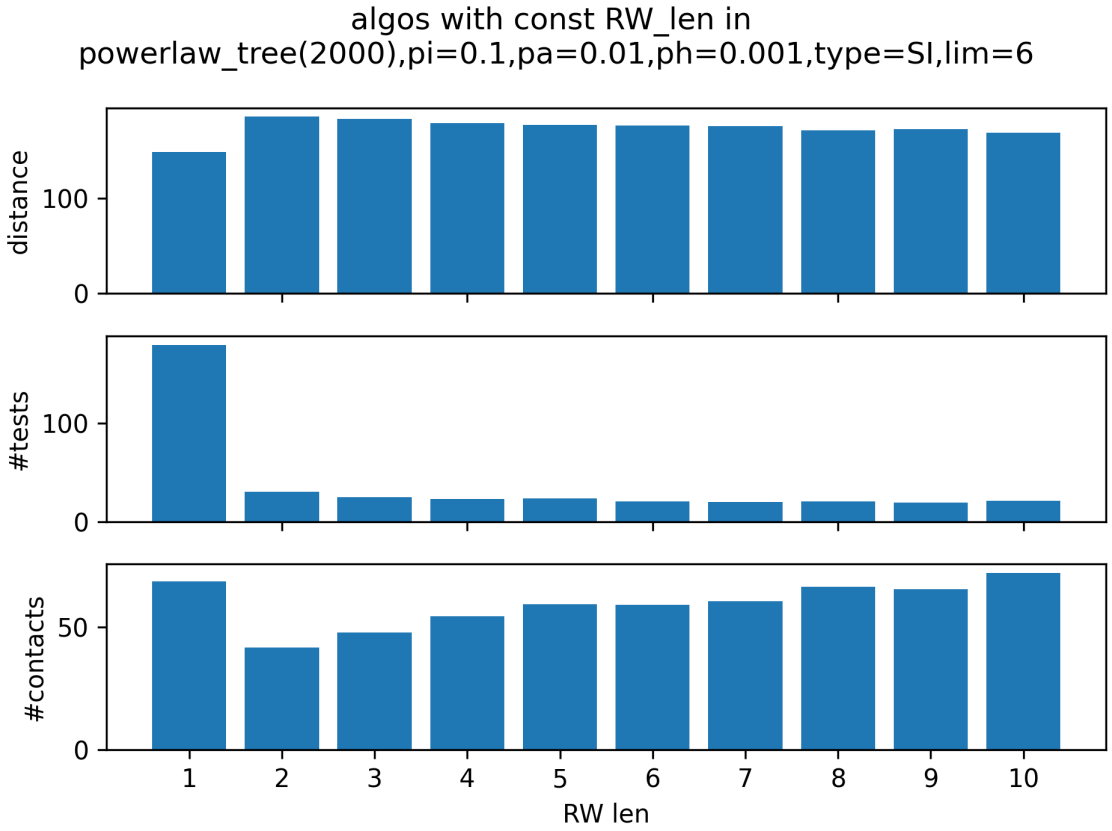

Figure 2: G is a power-law tree with 2000 nodes. By changing the length of the random walk from 1 to 2, the number of tests has dramatically decreased, and the number of contact queries has also decreased. Longer Random walks yield similar results with length 2, except that the number of contacts will slightly increase.

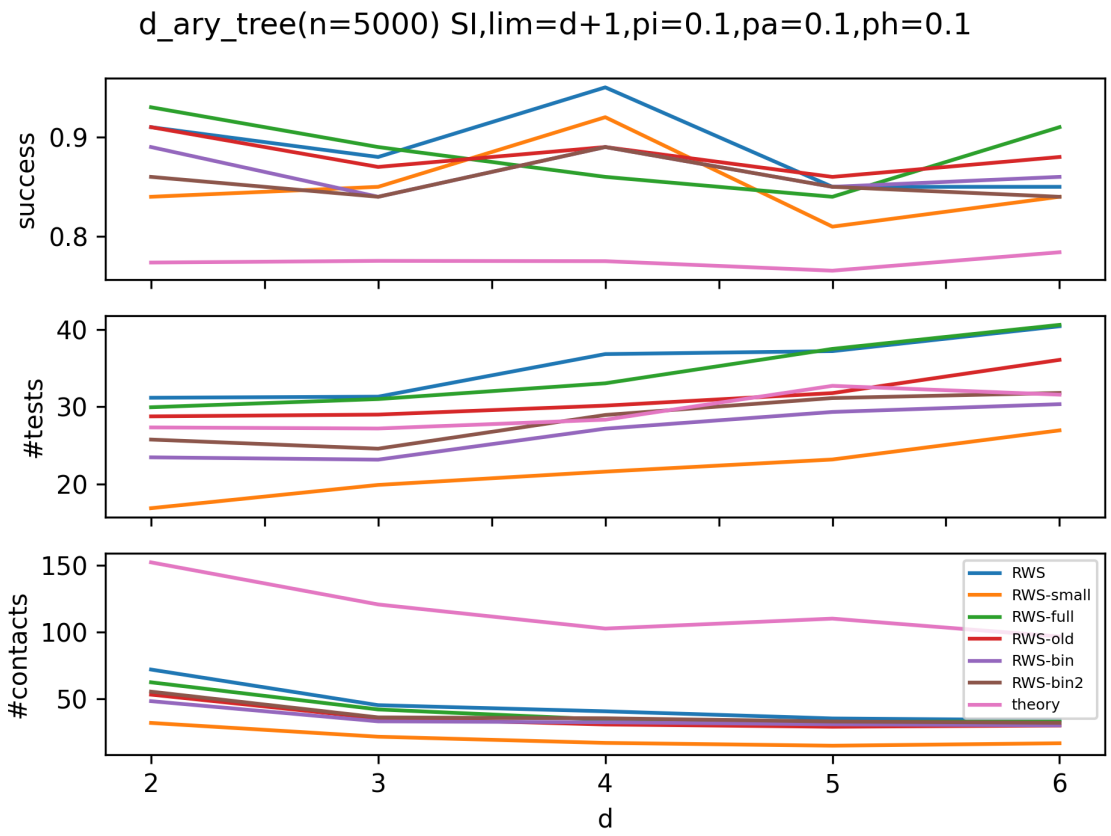

Figure 3: RWS on d-ary trees with 5000 nodes and $pa = 0.1$.

Figures 3 and 4 show the results of RWS on d-ary trees with 5000 nodes. In these figures, we can validate the theoretical results from Theorems 2, 3, and 4. The number of test and contact queries is linearly bounded in d. The theory line is present in these figures and shows the lower bounds for success and the upper bounds for the number of queries. The upper bound given for the number of tests is more realistic and closer to the actual numbers.

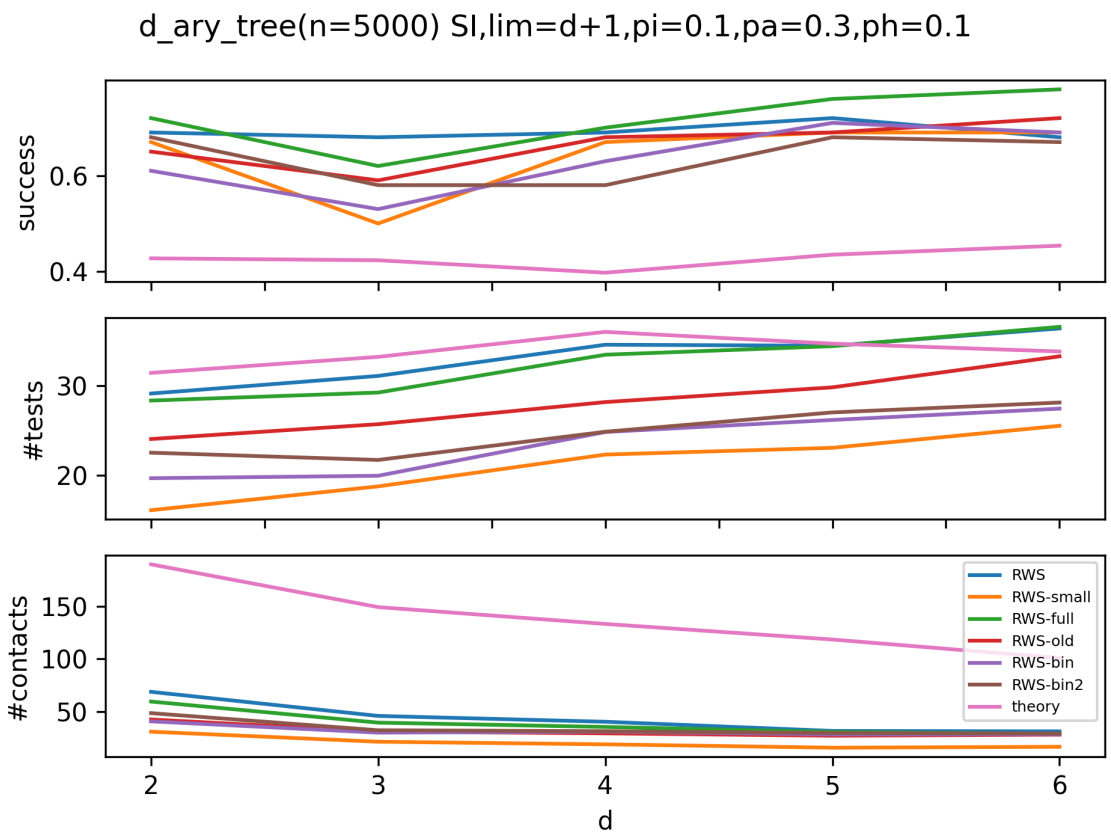

Figure 4: RWS on d-ary trees with 5000 nodes and $pa = 0.3$.

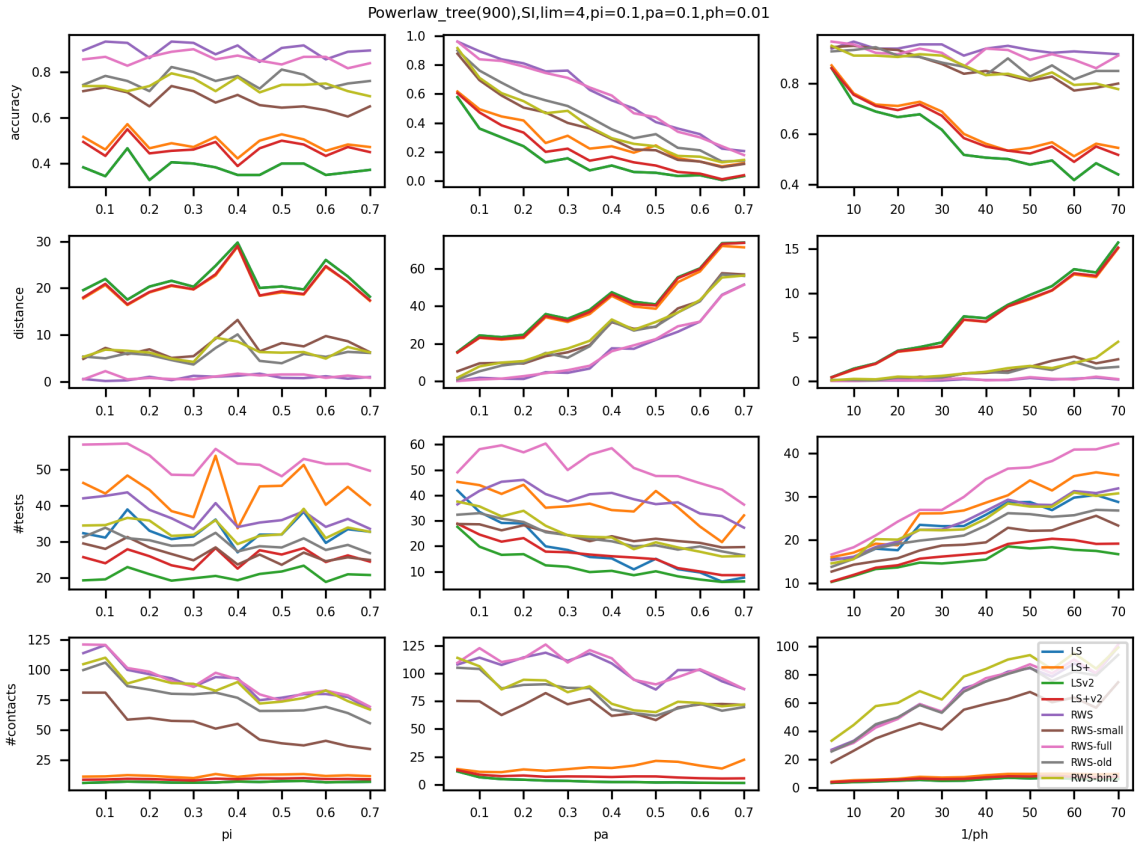

Figure 5: G is a power-law tree with 900 nodes. All four metrics are evaluated on RWS and LS algorithms.

In Figures 5 and 6, we can see that using RWS has better performance in terms of success probability and distance to the source for all versions of the algorithm. And number of tests in RWS is lower compared to LS+. The only drawback of RWS is the higher number of contact queries but it is much less expensive than test queries.

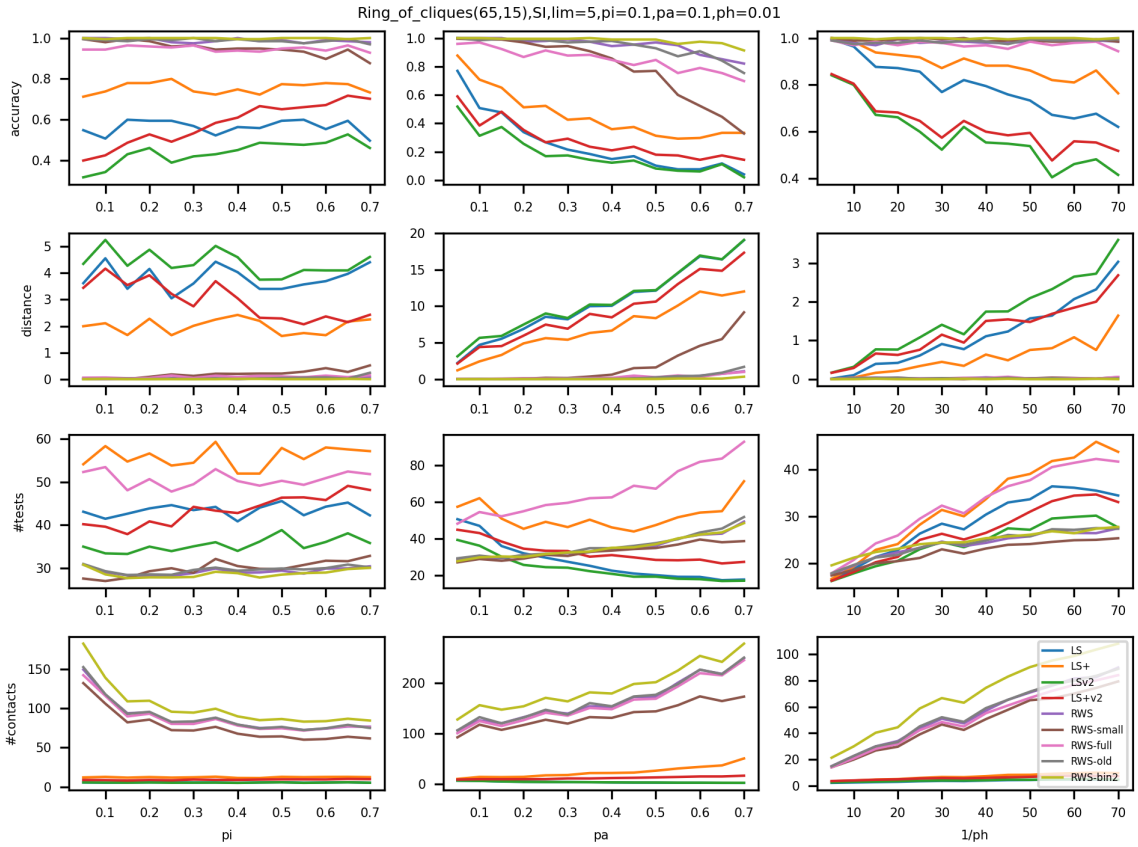

Figure 6: G contains 65 cliques with 15 nodes each. They are connected to each other via 15 single links. All four metrics are evaluated on RWS and LS algorithms.