Trial and Error in Influential Social Networks

Xiaohui Bei  Ning Chen  Liyu Dou
Division of Mathematical Sciences
Nanyang Technological University
Singapore

Xiangru Huang  Ruixin Qiang
Department of Computer Science
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

In this paper, we introduce a trial-and-error model to study information diffusion in a social network. Specifically, in every discrete period, all individuals in the network concurrently try a new technology or product with certain respective probabilities. If it turns out that an individual observes a better utility, he will then adopt the trial; otherwise, the individual continues to choose his prior selection.

We first demonstrate that the trial and error behavior of individuals characterizes certain global community structures of a social network, from which we are able to detect macro-communities through the observation of micro-behavior of individuals. We run simulations on classic benchmark testing graphs, and quite surprisingly, the results show that the trial and error dynamics even outperforms the Louvain method (a popular modularity maximization approach) if individuals have dense connections within communities. This gives a solid justification of the model.

We then study the influence maximization problem in the trial-and-error dynamics. We give a heuristic algorithm based on community detection and provide experiments on both testing and large scale collaboration networks. Simulation results show that our algorithm significantly outperforms several well-studied heuristics including degree centrality and distance centrality in almost all of the scenarios. Our results reveal the relation between the budget that an advertiser invests and marketing strategies, and indicate that the mixing parameter, a benchmark evaluating network community structures, plays a critical role for information diffusion.

Categories and Subject Descriptors
F.m [Theory of Computation]: Miscellaneous

General Terms
Economics

Keywords
Trial and error, social networks

1. INTRODUCTION

It is well-documented that information spreads via individuals’ interactions in social networks. The dynamic processes governing the diffusion of information and “word-of-mouth” effects have been studied extensively in various disciplines, e.g., Epidemiology, Sociology, Economics, and Computer Science. A central question studied in all of these disciplines is that how the dynamics of information diffusion unfolds within a social network.

Motivated from the fact that individuals in a social network are self-interested entities, applying game theoretical analysis to study the diffusion of information has received considerable attention in recent years [17, 10, 28, 29, 31, 22]. A common assumption in all of these game theoretical models is that an individual makes a rational choice to adopt a new strategy (e.g., a product) if it increases his payoff, which takes place when enough of his neighbors have adopted it. In other words, the spread of information is driven by strategic incentives.

While game theoretical analysis captures individuals’ self-motivated behavior, in many real social networks, people may have little knowledge about the structure of the underlying network, either globally or locally. Therefore, the interactions between individuals and adoptions of new products are rather blinded. Another important aspect ignored by the aforementioned models is that some individuals, endowed by their own personal characteristics, may have adventures to try new products. For instance, an iPhone user may try for curiosity a Samsung’s product and even adopt it if obtaining better experience. How does information diffuse in a social network given such (to some extent) bounded rational behavior of individuals? In this paper we will address this question by proposing a trial and error model to study the diffusion of information.

Trial and error is a heuristic method of problem solving and knowledge acquisition. It is often employed in incomplete information settings where limited guidance could be drawn from theory. In a trial and error process, people learn from both their successes and failures of occasional trials of new strategies and adapt their future decisions according to accumulated knowledge from experience. A large volume of literature has been accumulated on trial and error, spanning across different disciplines from various aspects, including, e.g., evolution of cognition [1], product develop-
ment [4], business innovation [26], game theory [30], and computational complexity [2], to name a few.

In the trial and error model introduced in the present paper, for every discrete period, individuals in a social network concurrently try a new product with certain respective probabilities. If an individual observes a better outcome, he will then adopt the trial; otherwise (i.e., upon an error), the individual continues to use his prior selection. The trial and error process is nondeterministic and concurrent, and creates a cascade of adaptive adoptions in a social network. Our model still assumes that individuals behave rationally, and beyond that, characterizes the aforementioned user behavior in a social network with incomplete information.

1.1 Our Results

In a real social network, people in the same community tend to adopt the same product. We first demonstrate that the trial and error dynamics exhibits the same phenomenon, that is, we are able to detect macro-communities through the observation of micro-behavior of individuals. Detecting communities is of great importance in disciplines where interactions are represented as graphs and has a variety of algorithmic challenges (see a survey by Fortunato [11]). Our results show that the natural trial and error behavior of individuals characterizes certain global community structures of a social network; this gives a solid justification of the model.

Specifically, we simulate the trial and error dynamics on the community detection testing graphs of Lancichinetti et al. [20], who described a procedure that generates random graphs with power law distributions on the degree of a node and the size of a community. The graphs generated by [20] are a refinement of the classic Girvan and Newman’s testing graphs [12] and has been taken as benchmarks for community detection algorithms. We compare the trial and error dynamics with the Louvain method [3], one of the most widely used modularity maximization approaches for community detection. Our simulation results show that the trial and error dynamics detects communities with high precision and even outperforms the Louvain method if individuals have dense connections within communities.

We next study the influence maximization problem [18]: Suppose that a new product is entering a market, which we assume that the network is connected. There is a weight function $w: E \rightarrow \mathbb{R}^+$ associated with each pair. That is, for any $(i, j) \in E$, $w(i, j)$ gives the tie strength between $i$ and $j$. In the present paper, we define $w(i, j) = |\{k \mid (i, k) \in E, (j, k) \in E\}| + 1$ which is the number of common friends of $i$ and $j$ plus one.\footnote{The concept of tie strength was first introduced by Granovetter in his landmark paper [13]; a basic hypothesis in [13] is that the stronger the tie between two individuals, the larger the proportion of individuals in the rest to whom they both are tied. In other words, there is a monotonically increasing relationship between weight and the number of common friends. Our definition of weights is one of the simplest functions that satisfy this hypothesis (note that the addition of one is to count the direct relationship between two individuals).}

Assume that there is a set of products $A$, and each vertex can choose one of the products. For each $i \in V$, let $f_i \in A$ denote the product that $i$ chooses; then $(f_i)_{i \in V}$ defines a configuration of the network. For any given configuration,
the normalized utility of agent \( i \) is defined as follows:

\[
    u_i = \frac{\sum_{j \in N(i), j \neq i} w(i, j) f(i, j) - f(i, j)}{\sum_{j \in N(i)} w(i, j)}.
\]

The numerator in the above formula is the total weight of the neighbors of \( i \) that choose the same product as him, and the denominator is the total weight of all edges incident to \( i \), which is a fixed number. The utility of an individual thus completely depends on his and his friends’ selections. In the present paper, we will consider a simple setting with only two products, i.e., \( A = \{ a, b \} \).

2.1 Trial and Error Dynamics

We will consider a trial and error model to study the dynamics of product adoption caused by self-interested behavior of individuals. Initially, every vertex is associated with a product. In each of the following rounds, all vertices will try another product concurrently according to the following probability function:

\[
    p(u) = e^{-\frac{u}{1-u}}.
\]

That is, assume that a vertex \( i \) chooses product \( a \) with utility \( u_i \) at the beginning of the round; then with probably \( p(u_i) \) the vertex will try another product \( b \), and with probability \( 1 - p(u_i) \) he will continue to use \( a \). If the vertex finds an improved utility at the trial \( b \) (with respect to the trials of all of his neighbors), then he will adopt the new product \( b \) at the end of the round. Otherwise (which corresponds to an “error”), he will continue to use the prior choice \( a \). The following figure shows the process of trial and error of a vertex within one round.

![Figure 1: Example of a non-stable Nash equilibrium](image)

We say a configuration is a zero configuration if all vertices have utility 0; otherwise we call it a nonzero configuration. It can be seen that zero configurations only occur in bipartite graphs where the two sides adopt different products. If the trial and error dynamics starts from a zero configuration, then at every round all vertices try another product with probability 1 and realize that it does not give a better utility (with respect to the trials of other vertices). Thus, all vertices will continue to choose their prior selections, i.e., the zero configuration will stay forever. We first establish the following characterization, which essentially eliminates zero configurations in the trial and error dynamics.

**Lemma 2.1.** A nonzero configuration will never move to a zero configuration in the trial and error dynamics.

2.2 Best Response Dynamics

In the context of game theory, our model can be considered as a game played on an underlying graph \( G = (V, E) \), where each pair of vertices \( i \) and \( j \) with \((i, j) \in E\) plays a coordination game. A profile (in our language, a configuration) is called a (pure) Nash equilibrium if no individual can improve his utility by unilaterally adopting another product. That is, from individuals’ strategic point of view, the dynamics of adoptions reaches a stable state. It is well-known that coordination games with players in an underlying graph are a potential game. Thus, a pure Nash equilibrium always exists, and can be achieved by the best response dynamics [24], i.e., individuals sequentially adopt products that yield the highest utilities.

The trial and error dynamics by its nature is quite similar to the best response dynamics. However, the trial and error dynamics allows concurrent moves for all individuals, whereas the best response dynamics considers sequential moves. If concurrent moves are allowed, the best response dynamics may not converge any more for many networks. For instance, for the complete graph with \( n \) vertices, initially, \( n/2 \) vertices choose product \( a \) and \( n/2 \) vertices choose product \( b \); then at every round all vertices change to adopt another product and the dynamics never converges.

The best response dynamics identifies the set of Nash equilibria as the set of stable configurations. But are all of these configurations really “stable” in a social network? Consider the example in Figure 1 in which there are a small clique \( G_1 \) with \( m \) vertices and a big clique \( G_2 \) with \( n \) vertices, and each vertex in \( G_1 \) connect to a vertex in \( G_2 \). Consider the special case when \( m = 2 \), i.e., \( G_1 \) contains only two vertices \( i \) and \( j \), and the configuration where \( i \) and \( j \) use product \( a \) while all vertices in \( G_2 \) use product \( b \). It is easy to verify that it is a Nash equilibrium, but such a Nash equilibrium rarely happens in a social network: the utility of both \( i \) and \( j \) is 0.5, and they are able to improve their utility to 1 if both adopting product \( b \). In other words, the small community can be easily influenced by the large community even if their connections are sparse.

2.3 Stable Nash Equilibrium

In this subsection, we will consider whether and when the trial and error dynamics stabilizes, and if yes, it stabilizes to which configuration. First, it is natural to see that a configuration that is not a Nash equilibrium should not be considered as stable: In such a configuration there always
exists a vertex whose utility is less than 0.5 and can be better if adopting the other product.

Second, not all Nash equilibria should be considered as stable. In the example of Figure 1, when the clique $G_1$ contains only 2 vertices, with probability $\left( e^{-\frac{0.5 - 0}{2}} \right)^2 = e^{-2}$ both of them will try and then adopt product $b$. That is, the Nash equilibrium configuration may jump to another configuration with a constant probability even in a single round; thus, the probability that such a configuration remains unchanged decreases exponentially as the dynamic process continues. However, if $G_1$ contains more vertices, then all vertices of $G_1$ have more utilities when adopting product $a$ because of internal edges and their trial probability becomes much smaller. Further, the only way to jump out of the Nash equilibrium configuration is to have at least half of the vertices of $G_1$ simultaneously try product $b$; this happens, however, with an extremely small probability. That is, while eventually the trial and error dynamics will change to another configuration as the process continues, this Nash equilibrium configuration will stay for a very long period.

Third, for some Nash equilibrium configuration, while there is a non-negligible probability to jump out, it is also very likely to return to the configuration. There are 2 small communities $G_1$ and $G_2$ which are connected by a vertex $k$ (see Figure 2). The initial adoptions are labelled next to each vertex. It can be seen that both $i$ and $j$ will try the other product with a constant probability in each round, but due to the dominance of their neighbors in the community, they will continue to choose the prior selection (i.e., $a$ and $b$, respectively). Now let us consider the following dynamics: both $i$ and $k$ try product $b$ (this happens with a constant probability), then $i$ will continue to choose $a$ but $k$ will adopt $b$. That is, the configuration is changed to a new one where the only difference is on vertex $k$. For the new configuration, consider another dynamics: both $j$ and $k$ try product $a$ (this happens again with a constant probability), then $j$ will continue to choose $b$ but $k$ will adopt $a$. That is, the configuration is changed back to the original one. It can be seen that the swaps between the two configurations will continue forever.

**Figure 2:** Example of swapping Nash equilibria

In summary, in a social network some configurations may stay for a long period or occur frequently. In both cases the configurations to some extent slow down or even block the diffusion of information, and thus, the network has reached a rather stable state. Given this observation, we next define the notion of $k$-stable Nash equilibrium, a refinement of the definition of Nash equilibrium, to characterize a stable state of a trial and error dynamic process.

**Definition 2.1.** For any given number $k$, we say that a trial and error process reaches a $k$-stable state if a Nash equilibrium configuration appears more than $0.5 \cdot k$ times in $k$ consecutive rounds. Such a Nash equilibrium is called a $k$-stable Nash equilibrium.

Note that the coefficient 0.5 in the definition can be replaced by any constant. This parameter together with $k$ characterize the extent of the stability of a Nash equilibrium. For instance, if $k = 1$ then the definition degenerates to the normal Nash equilibrium; and if $k = \infty$ then the only stable configurations are those that will stay forever (e.g., monopoly).

The theorem below shows that any trial and error process must reach a $k$-stable state for any finite $k$. However, different realizations of the process may reach different $k$-stable Nash equilibria. This is by the randomness of trial and error, and also reflects the nature of dynamics of information diffusion.

**Theorem 2.1.** For any finite number $k > 0$, the trial and error dynamics starting from a nonzero configuration reaches a $k$-stable state with probability 1.

In the following sections, we simulate the trial and error dynamics on various benchmark graphs and real social networks with vertex sizes ranging from 1000 to 25000. A general guidance for picking the best value of $k$ is that it should give us the most reasonable stable state of an underlying network. In all of our simulations, the trial and error process stabilizes to the first stable state within dozens of rounds, and then it stays at that configuration with very occasional local disturbances, for as long as $10^6$ rounds. We will therefore use $k = 10^6$ and consider the first $k$-stable state in all of our simulations. For convenience, in the rest of the paper, we will use stable state to denote a $k$-stable state.

### 3. Community Detection: Justification of Trial and Error

In the above section, we argued that the trial and error dynamics can lead to a more stable Nash equilibrium. In this section, we will provide a further justification to the trial and error model. Specifically, in practice, people in the same community usually turn out to adopt the same product. Does the trial and error dynamics exhibit the same phenomenon?

To answer this question, we simulate the trial and error dynamics to detect communities of a social network. We run two types of experiments on the benchmark graphs of Lancichinetti et al. [20] with 1000 and 5000 vertices, respectively. Similar to the setup of [20], we choose parameters $\gamma = 2$ and $\beta = 1$, where $\gamma$ and $\beta$ are the exponents of the power law distributions of the degree of a vertex and the size of a community, respectively. We choose the average degree of all vertices to be 15 and 20 for the 1000-vertex and 5000-vertex graphs, respectively, and the maximum degree to be 100. Further, every graph has an associated mixing parameter $\mu$, which denotes that each vertex shares a fraction of $1 - \mu$ of its edges with other vertices of its own community and a fraction of $\mu$ with the rest vertices of the graph. The generated graphs have well defined built-in community structures; more details of the benchmark graphs are referred to [20].

For every randomly generated graph, we run the following experiment: Initially every vertex chooses one of the products $a$ and $b$ with equal probability; we then simulate the
trial and error process from the initial state until it reaches a stable state. The simulation is repeated 100 times on the graph. For every pair of vertices \((i, j)\) that are connected by an edge, we denote by \(p(i, j)\) the probability that \(i\) and \(j\) end up with using the same product among all 100 simulations.

From the derived data, it can be seen clearly that when the mixing parameter \(\mu = 0.2\), all pairs of vertices are divided into two groups: one with small probabilities (roughly between 0.3 and 0.7) and one with probability 1. When \(\mu = 0.5\), the communities become fuzzy and there is no clear separation on the probabilities (there are positive densities for all probabilities larger than 0.3).

We next describe our community detection algorithm by the trial and error dynamics.

Algorithm 1 Community Detection by Trial and Error

1: Let \(V\) be the set of vertices of the graph and \(k = 1\).
2: Let threshold = 0.95.
3: while \(V \neq \emptyset\) do
4:   Pick \(i \in V\) and let \(S_k = \{i\}\).
5:   for \(j \in V\) do
6:     if \(p(i, j) >\) threshold then
7:       \(S_k \leftarrow S_k \cup \{j\}\).
8:   end if
9: end for
10: Let \(V = V \setminus V_k\) and \(k = k + 1\).
11: end while
12: Return \((S_1, S_2, \ldots)\).

In the algorithm, we first set a threshold value (which is 0.95 in the description below), and cluster vertices in the same community if their probabilities are beyond the threshold value.

We compare the performance of our algorithm with the Louvain method [3], which is a popular modularity maximization approach. Modularity maximization is one of the most widely used methods for community detection. The Louvain method iteratively optimizes local communities until global modularity can no longer be improved given perturbations to the current community state. To compare the built-in community structures of the generated graphs with our trial and error algorithm and the Louvain method, we use the normalized mutual information, which is a measure of similarity of partitions from information theory. The following figures show the comparison results (each point corresponds to an average over 10 graph realizations).

Notice that our algorithm detects communities with high precision (even outperforms the Louvain method) when the mixing parameter is less than 0.4. When the community structures become fuzzy as the mixing parameter approaches 0.5, the performance of our algorithm gradually drops. Note that the runtime complexity of our algorithm is rather high, as the probability \(p(i, j)\) for every pair is computed in terms of 100 simulations. However, here we do not attempt to propose a new community detection algorithm. Instead, the simulations indicate that the trial and error dynamics exhibit certain community structures of a social network. This gives a solid justification to the trial and error model.

4. INFLUENCE MAXIMIZATION

In this section, we study the word-of-mouth effect in our trial and error model. Given a social network \(G = (V, E)\), initially all vertices use product \(a\). A new product \(b\) now is about to enter the market. Given a sharp budget \(B\) on marketing, which vertices should be seeded in order to create a large fraction of adoptions? For each node \(i \in V\), let \(c(i) = \sum_{j, (i,j) \in E} w(i, j)\) be the cost to seed \(i\) using product \(b\) at the beginning. Let \(f(S)\) denote the expected amount of adoptions if \(S\) is the set of seeded vertices. Then the influence maximization problem is to find a set \(S \subseteq V\) to maximize \(f(S)\) given that \(\sum_{i \in S} c(i) \leq B\).

4.1 Algorithm

Our algorithm, at a high-level viewpoint, divides the network into communities and then considers each community separately. While we can use the community detection algorithms described in the previous section, how should we deal with each community? To answer this question, let us note that the best response dynamics does not have such community detection property; for instance, for the complete graph discussed earlier with \(n/2\) using product \(a\) and \(n/2\) using product \(b\), the best response dynamics does not even converge.

The cost to seed a vertex \(i\) here is defined as the total weight of all edges incident to \(i\). This definition is motivated from the observation that large degree vertices usually slow down the diffusion process [22]. Thus, seeding these vertices needs a larger cost.
first consider the special case where the underlying network is a complete graph; such a complete graph closely approximates the structure of a community. In a complete graph with \(n\) vertices, the weight of every edge is \(n - 1\) and thus \(c(i) = (n - 1)^2\) for all vertex \(i\). Since all vertices have the same cost and the same influence, the only (optimal) strategy is to seed \(\left\lceil \frac{n}{n-1} \right\rceil\) vertices arbitrarily. Note that the only Nash equilibria in a complete graph are the two monopolies where all vertices adopt the same product, either \(a\) or \(b\). Thus, the expected number of adoptions is equal to \(n \cdot \text{dominate}(k, n)\), where \(\text{dominate}(k, n)\) denotes the probability that product \(b\) dominates the entire market if \(k\) vertices are initially seeded.

The value of \(\text{dominate}(k, n)\) can be approximately computed via a straightforward dynamic programming procedure: Define \(\Pr[i][t]\) to be the probability that there are \(i\) vertices using product \(b\) at the end of round \(t\). Initially let \(\Pr[i][0] = 1\) and \(\Pr[i'][0] = 0\) for all other \(i' \neq k\). In general the value of \(\Pr[i][t]\) for each \(i\) can be computed precisely given \(\Pr[0][t-1], \Pr[1][t-1], \ldots, \Pr[n][t-1]\). Repeat the procedure until we have \(\Pr[0][t'] + \Pr[n][t'] > 1 - \epsilon\) for some \(t'\) and small enough \(\epsilon\), i.e., all vertices adopt the same product. Then \(\Pr[n][t']\) gives an approximation of \(\text{dominate}(k, n)\).

Figure 4a shows the results for several values of \(k\) and \(n\). We can see clearly that the probability that product \(b\) dominates increases rapidly when the fraction of targeted seeds goes from 50% to 60%, and the probability grows to 1 when the fraction is around 65%. In other words, seeding 65% vertices is sufficient to influence the entire graph and extra investment is redundant.

![Figure 4: Bang for the buck of complete graphs](image)

Figure 4: Bang for the buck of complete graphs

Note that a real social network in general exhibits strong community structures and vertices in the same community tend to use the same product. Given the connection between the trial and error dynamics and community structure established in the previous section, our community-based algorithm first partitions the network into disjoint communities, and then picks several communities and seed certain amount of vertices from the picked communities. While a network may not have perfect community structures in the sense that vertices may not be linked in the same community and there may have crossing-community edges, we expect (and are supported by the simulation results) that our analysis for complete graphs above gives close approximations to partitioned communities. A remaining question is which communities should be considered first in the algorithm.

Note that if we seed \(k\) vertices in a complete graph with in total \(n\) vertices, we spend \(k(n-1)^2\) amount of budget with an expected return of \(n \cdot \text{dominate}(k, n)\); thus, the efficiency of the investment, i.e., the expected number of vertices adopting product \(b\) per unit budget spent, is given by

\[
\text{invest}(k, n) \triangleq \frac{n \cdot \text{dominate}(k, n)}{k(n-1)^2}.
\]

Therefore, in order to spend the budget efficiently, we should always first consider those communities with larger \(\text{invest}\) values. Figure 4b shows the values of \(\text{invest}(k, n)\) for several choices of \(k\) and \(n\), from which we can conclude that: (1) for every \(n\), \(\text{invest}(k, n)\) achieves maximal at around \(k = 0.6 \cdot n\), and (2) the smaller a community is, the larger value \(\text{invest}(k, n)\) is.

Therefore, in the algorithm we always pick communities according to the increasing order of their sizes. The algorithm is described as follows.

**Algorithm 2 CommunityUp**

1. Let \((C_1, \ldots, C_k)\) be a partition of graph \(G\) based on some community detection algorithm.
2. Let \(\text{fraction}\) be the percentage of seeded vertices in a community.
3. Order communities by their sizes, e.g., \(|C_1| \leq |C_2| \leq \cdots \leq |C_k|\).
4. for \(\ell = 1, \ldots, k\) do
5. For the community \(C_\ell\), randomly pick \(\text{fraction} \cdot |C_\ell|\) vertices, and seed them if there is enough remaining budget.
6. end for

Notice that we use a parameter \(\text{fraction}\) instead of the nearly optimal value 60% in the description of the algorithm. In real social networks, a vertex may also be influenced by those from other communities, and the optimal value of the parameter \(\text{fraction}\) depends on the structure of the underlying network. In particular, if a network has poor community structures, one may need to seed a larger fraction of vertices in order to derive the same domination probability. In the simulations, we choose a slightly larger parameter \(\text{fraction} = 65\%\).

### 4.2 Simulations

We run a number of experiments to evaluate the performance of our community-based algorithm. We compare our algorithm with several other heuristic algorithms based on vertices’ structural measures of influence. Following the benchmarks considered in [18], we use degree and distance centrality heuristics as our comparison algorithms. Degree has long been considered as a standard estimate of a vertex’s influence in a network. Distance centrality, which is defined
as the total distance from one vertex to all other vertices, is another commonly used influence measure in sociology. It is based on the assumption that a vertex that has shorter distances to all other vertices has more chance to influence them.

Clearly seeding vertices with larger influences stands a better chance to influence more vertices. However, in our model a more influential vertex may have a larger cost. Thus, the fixed budget constraint yields a trade-off between the number of vertices one can seed and their influences. In our experiments, we consider two extremes of this trade-off: one always seeds vertices with larger influences and the other always tries to pick as many vertices as possible. Specifically, we run the following algorithms in the simulations.

- **CommunityUp**: Our proposed community-based algorithm. We use the Louvain method for community detection due to time efficiency, and select fraction = 65% as the percentage of seeding vertices in each community.
- **DegreeUp**: The degree greedy algorithm that seeds vertices in the order of decreasing degree until the budget is exhausted.
- **DegreeDown**: The degree greedy algorithm that seeds vertices in the order of increasing degree until the budget is exhausted.
- **DistanceUp**: The distance greedy algorithm that seeds vertices in the order of increasing distance centrality until the budget is exhausted. Note that some tested graphs are not connected. We then apply the approach of [18] by assigning a sufficiently large (e.g., the sum of all edge distances in the network) distance to all disconnected pairs of vertices.
- **DistanceDown**: The distance greedy algorithm that seeds vertices in the order of decreasing distance centrality until the budget is exhausted.

We choose the following two types of social networks as our testing data:

- The benchmark graphs of Lancichinetti et al. [20] with 1000 vertices, i.e., the same benchmark graphs used in the previous section for community detection. We pick two graphs with mixing parameter 20% and 50%, respectively. (We also run simulations for other scenarios with different mixing parameters and number of vertices, and obtain almost the same results.)

- A collaboration network denoted by CondMat from the e-print arXiv that covers scientific collaborations between authors with papers submitted to Condense Matter category. The CondMat network contains 23133 vertices and 186936 edges. Each vertex in the networks represents an author, and an edge between i and j indicates that there is at least one paper co-authored by i and j (multiple co-authored papers by two authors only count one edge). It is well-known that collaboration networks capture many of the key features of general social networks [23].

To compare simulation results on different graphs, we assume for convenience that the budget B is of a proportion $\lambda$ of the total cost of all vertices, i.e., $B = \lambda \cdot \sum_{i \in V} cost(i)$. We run all of the aforementioned algorithms on the testing networks with $\lambda = 1\%, 3\%, 5\%, 10\%, 20\%$, up to 70%, respectively. For each specific setting, we simulate the trial and error dynamics 100 times and take the average result.

![Figure 5: Results for the benchmark graph of [20]](http://snap.stanford.edu/data/CA-CondMat.html)

Figure 5 shows the performances of different algorithms in the benchmark graphs of Lancichinetti et al. [20]. Our community-based algorithm CommunityUp outperforms all other algorithms by a big margin. This shows that taking community structures into account can reduce the amount of inefficient investment and greatly improve the marketing result. A more detailed analysis shows that in the trial and error process with the CommunityUp algorithm, almost all communities picked by the algorithm are dominated by the new product in the stable state. For all other algorithms, however, there are a number of communities to which the algorithm invests a fairly amount of budget, but eventually all vertices inside still use the original product in the stable state.

Figure 6 shows the results of the CondMat collaboration network. The structure in this real social network is more complicated than the benchmark graphs. As a result, the improvement margins between CommunityUp and other algorithms are not as large as those in the benchmark graphs.

### 4.3 Discussions

From the simulations we can derive the following observations and conclusions.
degree or a large distance centrality, the edges incident to the
ios: Many companies tend to market products with small
some cases. Such phenomena echo some real world scenar-
更好的 than the ones that prioritize high-influence vertices
rithms that prioritize low-influence vertices perform much
also plays different roles in different algorithms. The algo-
community structures and outperform the other algorithms.
algorithm can still take advantages from such fuzzy
the budget keeps growing to around 70%. Yet our
budget goes from 40% to 60%, and then slowly goes to 1 as
the performances of some heuristic algorithms on such a
random strategy on a complete graph: the fraction of ver-
influences from vertices of other communities. As a result,
approximately straight lines in Figure 5a.

Low budget vs. high budget. The amount of budget
also plays different roles in different algorithms. The algo-
the ones that prioritize high-influence vertices when the budget is small. As the budget increases, the lat-
tter start to catch up and even outperform the former in
some cases. Such phenomena echo some real world scenarios:
the strategy is to sign celebrities who have large influences as
their spokespersons.

Mixing parameter. When the mixing parameter of
a network is small, each community in the graph is more or
less independent to each other. Thus, the number of com-
adopting the new product should be roughly pro-
portional to the budget invested, which is confirmed by the
approximately straight lines in Figure 5a.
Meanwhile, networks with large mixing parameters have
bad community structures. Each vertex would receive more
influences from vertices of other communities. As a result,
the performances of some heuristic algorithms on such a
graph (Figure 5b) become very similar to that of a purely
random strategy on a complete graph: the fraction of ver-
tices adopting the new product increases rapidly when the
budget goes from 40% to 60%, and then slowly goes to 1 as
the budget keeps growing to around 70%. Yet our Commu-
nityUp algorithm can still take advantages from such fuzzy
community structures and outperform the other algorithms.

Testing vs. real networks. When comparing the bench-
mark graphs of [20] with the real network CondMat, one can
see a noticeable similarity between Figure 5a and Figure 6.
In fact, the average mixing parameter of the CondMat net-
work is 0.289, which indicates that the CondMat network
has a pretty good community structure. On a side note, it
also shows that mixing parameter, a benchmark evaluating
whether a network has good community structures, is essen-
tial for the performances of different influence maximization
algorithms.

5. CONCLUDING REMARKS

The diffusion of information in a social network is affected
not only by its network structure but also by individuals’ preferences and experiences. We introduced a trial and error model to study the dynamics of information diffusion due to strategic incentives of individuals. We showed that the trial and error dynamics reveals certain global community structures of a social network and from which we are able to detect macro-communities through the observation of micro-behavior of individuals. We also studied the influence maximization problem and proposed a heuristic algorithm based on community detection. Simulations showed that the algorithm significantly outperforms several well-studied heuristics including degree and distance centralities.

We believe that investigating trial and error as the bridge linking individuals’ behavior in the micro-level and social phenomena in the macro-level would bring us a deeper understanding of how information diffuses in a social network. Our work leaves a number of intriguing questions and directions for future studies.

- In our work, we considered only two products. It is
  natural to extend the model to more generalized settings
  with more products. It is also interesting to consider
  other probability and edge weight functions.

- The main focus of our study is to investigate the conse-
quency of trial and error behavior in social networks
assuming homogenous individuals, and we only utilize
undirected social networks in the current analysis. It is
argued in [14] that social relationships could be asymmetric as the LinkedIn network. Thus, for some settings it is more appropriate to use a directed network to model the asymmetric influences among individuals.

- A key question in the study of information diffusion is the rate of convergence. For epidemic models, a major prediction is that information spreads quickly in highly connected networks [9], whereas for game theoretical models, it was argued that information spreads quickly in locally connected networks [22]. What is the rate of convergence in our trial and error model?

- Another important individuals’ behavior is the noisy
best response dynamics where instead of always choos-
ing the best response strategy, there is a small per-
turbation for individuals to choose a different strategy
even though it may give them a worse payoff. Noisy
best response has been studied extensively in [15, 17,
28, 29], with a focus on the stationary distribution of
the corresponding Markov chain. A main result along
this line is that the noisy best response dynamics al-
ways converges to a particular equilibrium in which a
product dominates the market.

We may consider a similar noisy trial and error model:
even though an individual has utility 1, he still has a
nonzero probability to try a new product, and adopt it
if obtaining a larger utility (which occurs when some
of his neighbors also make a trial). We can show similarly that the noisy trial and error dynamics also always reaches a monopoly equilibrium. In contrast to the noisy best response dynamics, the perturbation in our model is purely at trials and all individuals are still completely rational (i.e., they never adopt a product that yields a smaller utility). All our simulation results carry to the noisy trial and error dynamics. Therefore, while it may need an extremely long period to reach a monopoly equilibrium, the dynamics arrives quickly at a fairly stable state which allows coexistence of both products. Indeed, this is precisely what we observe in a real marketplace where in most cases several competing products survive. An intriguing question is to theoretically characterize stable state and stable Nash equilibrium in the (noisy) trial and error model.

6. REFERENCES