Simpler Sybil-Proof Mechanisms for Multi-Level Marketing

FABIO A. DRUCKER and LISA K. FLEISCHER, Dartmouth College

Multi-level marketing refers to a marketing approach in which buyers are encouraged to take an active role in promoting the product. This is done by offering them a reward for each successful referral of the product to other prospective buyers. To encourage potential customers to buy early and to give referrals to influential people, these mechanisms also reward *indirect* referrals — a direct referral linked to the buyer through other direct referrals. Doing so can make the referral/reward system vulnerable to sybil attacks — where profit maximizers create several replicas in order to maximize their rewards.

In this paper we propose a family of mechanisms for which sybil attacks are not profitable. We do this by modifying any mechanism that satisfies certain natural properties of sensible reward mechanisms to obtain one that is invulnerable to *sybil attacks* by profit maximizers while preserving its natural properties. Our modified mechanisms are also collusion proof. Finally, we give a concrete example of a natural mechanism that is sybil proof and simple to implement.

Categories and Subject Descriptors: J.4 [**Social and Behavioral Sciences**]: *Economics*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Network problems*

General Terms: Algorithms, Economics

Additional Key Words and Phrases: Mechanism design, Recommender systems, Social networks

1. INTRODUCTION

Multi-level marketing refers to a marketing approach in which buyers are encouraged to take an active role in promoting the product. This is done by offering them a reward for each successful referral of the product to other prospective buyers. To encourage potential customers to buy early and to give referrals to influential people, these mechanisms also reward *indirect* referrals — a direct referral linked to the buyer through other direct referrals. For example, if Alec refers Betty and then Betty refers Carol, Alec is rewarded for both Betty's and Carol's purchases.

This corresponds to a scenario in which an online merchant is selling a product. After purchasing, each buyer can then choose to refer people they know. The seller wants to offer enough rewards to encourage referrals to occur. However, the underlying social network is initially unknown to the seller, who can only observe purchases and referrals. There is no cost for making referrals, but each buyer is only willing to make a limited number of referrals. Thus, the seller wants to encourage buyers to make their limited referrals to *influential* people that can then refer many people.

Product referrals can be modeled using a directed *referral tree* T. Each node in T corresponds to a buyer. Tree T has a directed edge from v to u if u buys the product as a

6211 Sudikoff, Dartmouth, Hanover, NH, 03755, USA. Email: {druckerf,lkf}@cs.dartmouth.edu.
Supported in part by NSF grants CCF-0728869 and CCF-1016778. A one page abstract partially summarizing these contributions appeared in the Workshop on the Economics of Networks, Systems and Computation [Drucker and Fleischer 2012]. Full version of this paper available at www.cs.dartmouth.edu/∼druckerf/papers/sybil-proof.pdf.

EC'12, June 4–8, 2012, Valencia, Spain.

result of a referral from v. A *reward mechanism* takes a referral tree T and determines the reward, $R(v)$, each node $v \in T$ gets.

One potential drawback to multi-level mechanisms is that they could allow for sybil attacks. A *sybil attack* consists of a user purchasing multiple copies of the product under false identities to increase their reward. For example, a user can create two identities, the second with a referral from the first. Since the underlying social network is unknown to the seller, he may not be able to detect the creation of these false identities. For each referral made through the second identity, both identities capture some reward. Under some conditions this increases the attacker's reward by more than π, the unit price of the product, thus increasing his profit. In general, a node v can perform a sybil attack by replacing itself in the tree with l *replicas*. After the attack is performed, each of the children of v is the child of one of these replicas. However, any two children of v could now have different replicas of v as parents, and it is possible for one replica of v to have other replicas of v as children.

A sybil attack is undesirable for two reasons:

(1) It directly reduces the profit of the seller, since buyers are getting higher rewards than what the seller intended for them to receive.
(2) After v performs a sybil attack, ancestors of v are farther from the descendants of v in the referral tree. Under many reward mechanisms, this means ancestors of v get less reward from descendants of v. The parent of v may thus decide to refer a less influential node instead of referring v, since less influential nodes have less incentive to perform sybil attacks, and therefore yield more reward to their ancestors. But, less influential nodes refer fewer people, so they are less profitable for the seller.

Thus, sybil attacks can reduce the seller's profit it two ways. In Section 3.1, we show specific examples of both effects for an example reward mechanism.

To analyze the effects of sybil attacks, it is helpful to think in terms of potential buyers who have intrinsic value for at most a single unit of product and would purchase additional units only to benefit by receiving additional rewards. [1]

The *profit* $P(U)$ of a set of nodes U is the total reward received by the nodes in U minus the cost of purchasing the product incurred by those nodes: $P(U) := \sum_{u \in U}(R(u) - \pi)$.

A sybil attack is *profitable* if it increases the profit of the node performing it. A tree T' is obtained by a *split* from T at v if T' can be the result of $v \in T$ performing a sybil attack. A reward mechanism is *sybil-proof* if no sybil attack can be profitable under it: that is, if for any trees T, T', if T' can be obtained from T by a split at v, then the reward of v in T' is not higher than its reward in T.

The following are some natural properties of reward mechanisms:

(1) *Budget constraint*: The seller is only willing to spend a maximum amount $C < \pi$ per purchase in rewards. Therefore, the total reward paid should satisfy $\sum_{u \in T} R(u) \leq |T|C$.
(2) *Subtree constraint*: $R(u)$ should depend only on the subtree rooted at u. Intuitively, the reward of u should not depend on u's position in the tree, since u should not have an incentive to delay buying the product to hold out for a referral in a more rewarded position in the tree.
(3) *Monotonicity*: If u is a descendant of v, adding a child to u increases v's reward at least as much as adding a child to a descendant of u. Monotonicity limits the scope of indirect rewards: while indirect referrals are rewarded, direct rewards trump in-

[1] We show in Section 5 that our results extend to buyers with value for more than one item.

direct rewards. This encourages the formation of short chains since it concentrates rewards for a given purchase in nodes close to it. This is desirable because successful recommendation chains tend to be short for most products [Leskovec et al. 2007a].

(4) Fairness, expressed as *anonymity*: if we replace a node $v \in T$ with a node u, and set all children of v to be children of u instead, the reward of u is the same as the reward of v was, and the reward of all other nodes in T remains the same.

We also discuss the following properties:

(5) Unbounded rewards: Potential rewards are unbounded even given a limit d in the number of referrals each person can make. That is, there is a fixed number d such that for any number $r \in \mathbb{R}^+$ there exists a tree rooted at u of maximum degree d such that $R(u) \geq r$. The possibility of unbounded rewards incentivizes buyers to refer influential people, since it allows for potentially unlimited rewards even if each buyer makes only a limited number of referrals.

(6) Summing contributions: There exists a sequence $\{c_k\}_{k\geq 1}$ such that given any tree T_v rooted at v, $R(v) = \sum_{u \in T_v} c_{\text{dist}(v,u)}$, where $\text{dist}(v, u)$ is the length of the shortest path from v to u. A mechanism that satisfies summing contributions necessarily satisfies the subtree constraint.

One example of an anonymous reward mechanism that satisfies the subtree constraint, the budget constraint, unbounded rewards, monotonicity, and summing contributions is the geometric reward mechanism, described in Section 3. However, this mechanism is not sybil-proof. Emek, Karidi, Tennenholz, and Zohar [Emek et al. 2011] give a simple anonymous mechanism that is "locally" sybil-proof. A *local* sybil attack is a sybil attack where all the children of the original node are children of the same replica. A reward mechanism is *locally sybil-proof* if no local sybil attack can be profitable under it. The local sybil-proof mechanism in [Emek et al. 2011] is vulnerable to more sophisticated sybil attacks. They also give a complicated sybil-proof mechanism, but it is not clear how to implement this more complicated mechanism in polynomial time. Aside from its complexity, the sybil-proof mechanism in [Emek et al. 2011] suffers from another drawback: it does not satisfy monotonicity, since it ignores large portions of the referral tree. So, "users that recruit a large number of buyers may find that they are not rewarded for many of them" and it is "far from being intuitive" [Emek et al. 2011]. This could reduce the incentive for buyers to promote the product. For example, it is possible that a node's reward could be unaffected by the addition of an extra direct referral but to increase with the addition of an indirect referral (see full version).

Our Contribution. In this paper, we give simple and natural mechanisms that are sybil-proof and easy to implement. We give a simple way of modifying any anonymous reward mechanism that satisfies summing contributions, monotonicity, and the budget constraint to obtain an anonymous mechanism that satisfies the subtree, monotonicity, and the budget constraint, and is sybil-proof. If the original mechanism satisfies unbounded rewards, so does the modified mechanism (Section 3). Also in Section 3, we show that there is no anonymous sybil-proof mechanism that rewards indirect referrals and satisfies summing contributions. We prove that our modified mechanisms are sybil-proof (Section 4). Next, we show that our mechanisms are resilient to sybil attacks even if an agent has utility for multiple units of products, or if a group of agents jointly performs an attack. That is, our mechanisms are collusion sybil-proof (Section 5). Finally, we give a concrete example of a sybil-proof mechanism that satisfies the subtree and budget constraints, monotonicity, unbounded rewards, and is simple to implement (Section 6).

All the above results hold for a more general *graph setting* in which the referral credit for each purchase can be divided among multiple referrers. In Section 2 we formalize this more general model. Throughout the paper we show that all our results hold in this more general setting.

The main idea behind our sybil-proof mechanism is the following: First, we show that if it is possible for a node to get a reward of more than π from a single child, then the mechanism cannot be sybil-proof. Our modification caps at π the reward a node v can get through any single tree rooted at a child u of v. If v was getting a reward of $\pi + F$ from the tree rooted at u, this reduces the reward of v by F. It then recursively reassigns this reward of F to the tree rooted at u. This shifts as little reward as possible from the ancestors of a subtree to the subtree while rendering the mechanism sybil-proof.

Our modified mechanism maximizes the reward a node gets subject to the constraint that the same budget per purchase is maintained and the reward of its ancestors is reduced the minimum necessary to render the mechanism sybil-proof. This makes it possible for buyers that refer at least two people to get potentially unbounded rewards, therefore providing incentive to refer people that are as influential as possible. This leads to more referrals and therefore increases the seller's profit.

We make no assumptions about the underlying structure of the social network, the number of referrals each buyer is willing to make, or the way in which referrals affect the probability of a purchase occurring. This allows a seller to chose the most appropriate mechanism from those that satisfy the stated properties according to their own model of these elements for their specific case, and modify it in the way we describe to obtain a sybil-proof mechanism.

Shifting rewards around the tree may impact the decisions about whether to make referrals, so some referrals that would occur with the unmodified version of our mechanism may not occur in the modified version. However, our proposed modification seeks to minimize this effect by modifying the original mechanism as little as possible. Additionally, this effect could partly be counteracted by the fact that our modification gives more reward to nodes closer to referrals, so it could also lead to some referrals that would not occur with the original mechanism.

Other related work. [Abbassi and Misra 2011] consider the problem of designing a mechanism for a setting almost identical to ours, and introduce the graph model we explain in Section 3. They study a model in which each user makes a limited number of referrals. Each node has a randomly chosen threshold t, and only refers users if their expected reward from the referral is greater than t. They assume nodes know what the structure of the social network is, and that nodes that get a referral purchase the product with some probability p. They propose a reward mechanism where the reward for each purchase is divided equally between the $k-$closest ancestors to the purchasing node, where k is a parameter to their mechanism. They show via simulations on graphs obtained from real social networks that increasing k from 1 up to the diameter of the graph monotonically increases the expected number of users that hear about the product. They do not consider the problem of sybil attacks, and their proposed mechanism is not sybil-proof and does not satisfies the subtree constraint.

Because of its relevance to a wide array of problems, including the spread of epidemics, new ideas, or the adoption of new products and behaviors, there is much work investigating how innovations spread through a network. Existing work in computer science and economics is summarized in [Kleinberg 2007]. This problem has also been studied in other disciplines. Granovetter [Granovetter 1978] takes a sociological approach to the problem of how a new behavior spreads through a social network, to

try to explain seemingly paradoxical situation in which collective behavior is divorced from individual preferences.

Recent work considers a problem in which agents on a network offer a reward for the first successful answer to a query. The main difference between this problem and our own is that rewards are generated only for the first node to successfully answer a query, while in our case rewards are generated by every node in the tree. A desirable reward schemes encourages nodes that receive the query to transmit it to their neighbors. However, when a node transmits a query the number of nodes trying to answer it increases. This reduces the probability that the transmitting node is the one to successfully answer it and get the reward. Thus, indirect rewards are needed to overcome this incentive and encourage nodes to transmit queries. This problem is very similar to the approach to DARPA's red balloon challenge taken by the winning team [DAR 2009; Pickard et al. 2010]. The solution proposed in [Pickard et al. 2010] is vulnerable to sybil attacks; [Babaioff et al. 2011] proposes an alternative solution that is resilient to them. Another approach to this problem is proposed by [Cebrian et al. 2011; Kleinberg and Raghavan 2005]: each node, upon receiving an offer for a successful answer to a query, decides how much to offer its neighbors for a successful answer.

Domingos and Richardson [Richardson and Domingos 2002] introduce the problem of using data mining to determine the value of each potential buyer in a social network, allowing a viral marketer to determine how much should be spent to directly market to a particular individual. Kempe, Kleinberg and Tardos [Kempe et al. 2003] and Leskovec el al. [Leskovec et al. 2007b] explore this question for a variety of network models. The related problem of determining how to divide a fixed budget among nodes in a network to maximize the spread of product through the network is studied in [Douceur and Moscibroda 2007] and [Singer 2012]. Other work focuses on the related problem of determining marketing and pricing strategies to maximize the seller's profit in a social network, taking advantage of the influence earlier purchasers effect on their neighbors [Arthur et al. 2009; Hartline et al. 2008]. These approaches assume that the social network is known by the seller in advance, which precludes the possibility of sybil attacks.

2. THE MODEL

Tree setting. We model referrals with a directed tree T. Nodes in T correspond to buyers, and there is an edge from v to u in T if u's purchase is a result of a referral from v. For a tree T, let T_v be the subtree rooted in v. Let δ_v^+ be the set of all children of v. A vertex u is a *descendant* of v if $u \in \{T_v \setminus \{v\}\}$. If node u is a descendant of v, dist(v, u) is the length of the path from v to u. Node u is the parent of v, denoted parent(v), if there is an edge from u to v. Vertex v is an *ancestor* of u if $u \in T_v$. Node u is a *direct referral* of v if u is a child of v. Node u is an i*-indirect referral* of v if $u \in T_v$ and dist$(v, u) = i$.

Graph setting. The tree setting assumes that a buyer's purchase is the result of a single referral. However, somebody's decision to purchase could be influenced by multiple people. Maybe u decides to purchase only after receiving recommendations by all of v_1, v_2 and v_3. In this case, the reward for u's purchase could be split among v_1, v_2 and v_3 in some way chosen by u.

We model this scenario with directed acyclic *referral graphs*. In a referral graph $G = (V, E)$, there is a directed edge from v to u if u's decision to buy is partly a result of v's referral. Each edge $e = (v, u)$ has a weight $\alpha_e = \alpha_{vu}$. The weight α_{vu} corresponds to the fraction of the credit for u's purchase that is assigned to v.

For a graph $G = (V, E)$, let G_v be the subgraph rooted at v. Let δ_v^+ be the set of all children of v; that is $\delta_v^+ := \{u \in G : (v, u) \in E\}$. Let δ_v^- be the set of all parents of v; that is $\delta_v^- := \{u \in G : (u, v) \in E\}$.

Let $\mathcal{P}_{vu}$ be the set of all paths from v to u. Note that $p_1, p_2 \in \mathcal{P}_{vu}$ may not be disjoint. The set $\mathcal{P}_{vv}$ contains only the path p_{vv} consisting solely of vertex v. The *weight* of a path p, $\alpha(p)$ is the product of the weights of all edges along that path: $\alpha(p) := \prod_{e \in p} \alpha_e$. The weight $\alpha(p_{vv}) = 1$ for all v. The *length* of a path p, $|p|$, is the number of edges in p. With respect to G_v, the *weight* of node $u \in G_v$, $\alpha_v(u)$, is the sum of the weights of all paths from v to u: $\alpha_v(u) := \sum_{p \in \mathcal{P}_{vu}} \alpha(p)$. The weight of a graph G_v is the sum of the weights in G_v of all the nodes in G_v: $\alpha(G_v) := \sum_{u \in G_v} \alpha_v(u)$.

A vertex u is a *descendant* of v if $u \in \{G_v \setminus \{v\}\}$. Vertex v is an *ancestor* of u if $u \in G_v$. Node u is a *direct referral* of v if u is a child of v. Node u is an *i-indirect referral* of v if $u \in G_v$ and there is a path of length i from u to v.

$R(G_v)$ refers to the combined reward of all nodes in G_v proportional to their weight in G_v. So, $R(G_v) := \sum_{u \in G_v} \alpha_v(u) R(u)$, and $R(G_v \setminus \{v\}) := \sum_{u \in G_v \setminus \{v\}} \alpha_v(u) R(u)$.

Since the weight α_{vu} corresponds to the fraction of the credit for u's purchase that is credited to v, the sum of the weight of all incoming edges to u in G is 1 for every non-root node u. That is, for every $v \in G$, $\sum_{v \in \delta_u^-} \alpha_{vu} = 1$. Thus, for all v, u, $\alpha_v(u) \leq 1$. A subgraph G_v may not include some edges to u, so the sum of the weights of the edges into u in G_v may be less than 1.

The subtree and budget constraints, anonymity, and unbounded rewards as defined for the tree setting extend naturally to the graph setting. When referring to the graph setting, we rename the subtree constraint as the *subgraph constraint*, and we interpret the bound d on the degree of a tree to apply to the out-degree of nodes. Extending summing contributions and monotonicity is more complicated. We now define those properties for the graph setting.

Suppose we compare adding a new node to a graph G as a child of u, which is in turn a descendant of v, or as a child of w, a descendant of u. There may be paths from v to w that do not go through u, and these may be shorter or have greater weight than paths from v to u. If so, a mechanism may want to reward v more for a child of w than for a child of u: w may be a "closer" descendant of v than u despite u being an ancestor of w. Thus, monotonicity as defined for the tree setting is no longer a natural or desirable property in general.

However, in the special case in which all paths from v to w do go through u, adding a node as a child of u should reward v at least as much as adding it as a child of w. So, in the graph setting, we define monotonicity as: For all $u \in G_v$, and all $w \in G_u$, if all paths from v to w go through u, then adding an edge with weight α' from u to any node x increases the reward of v at least as much as adding an edge with the same weight α' from w to x instead. If G is a tree, this definition reduces to the definition of monotonicity for the tree setting.

An extension of summing contributions to the graph setting should account for the fact that there may be paths from v to u of different length. The reward v gets from u should depend on the lengths and weights of all paths between them. Then, given a sequence $\{c_k\}_{k \geq 1}$, we denote the *contribution* a node v gets from u as $c(v, u)$. This is the total reward v gets from u's purchase, summing over all paths from v to u. So $c(v, u) := \sum_{p \in \mathcal{P}_{vu}} c_{|p|} \alpha(p)$. Then in the graph setting, a mechanism satisfies summing contributions if there exists a sequence $\{c_k\}_{k \geq 1}$ such that given any graph G_v rooted at v, $R(v) = \sum_{u \in G_v} \sum_{p \in \mathcal{P}_{vu}} c_{|p|} \alpha(p)$. In the tree setting, $|\mathcal{P}_{vu}| = 1$. Since all edges have weight 1, the weight of the path p_{vu} from v to u is 1, and $|p_{vu}| = \text{dist}(v, u)$. So this

definition reduces to our definition of summing contribution for the tree setting when G is a tree and all edges have weight 1.

3. THE MECHANISM

In this section we present an example of a simple anonymous reward mechanism that satisfies summing contributions, monotonicity, the budget constraint, and unbounded rewards. We show that this mechanism is vulnerable to sybil attacks, and in fact we prove that no mechanism that satisfies summing contributions can be sybil-proof and reward indirect referrals. Then, we introduce a simple way to modify any anonymous reward mechanism that satisfies summing contributions, monotonicity, and the budget constraint to obtain an anonymous mechanism that satisfies the subtree, monotonicity, and budget constraints, and is sybil-proof. We show that if the original mechanism satisfies unbounded rewards, so does the modified mechanism.

3.1. The geometric reward mechanism

The geometric reward mechanism is a simple mechanism that satisfies summing contributions, the budget constraint, monotonicity, and anonymity.

In the *geometric reward mechanism* for the tree setting, node v receives ab for each direct referral and $a^i b$ for each i-indirect referral, for all $i \geq 1$. To guarantee that the budget constraint is met, a and b must also satisfy $\sum_{i=1}^{\infty} a^i b \leq \phi\pi$; so that $b + \phi\pi \leq \frac{\phi\pi}{a}$. We use $R^g(v)$ to denote the reward node v gets in the geometric mechanism. This mechanism satisfies summing contributions with $c_i = a^i b$. In the graph setting, the geometric mechanism is defined by $R^g(v) = \sum_{u \in G_v} \sum_{p \in \mathcal{P}_{vu}} a^{|p|} b\alpha(p)$.

Barring sybil attacks, the geometric mechanism also satisfies the unbounded reward constraint. Let T_v be a tree where each node at distance $< n$ from v has $d = \lceil 1/a \rceil$ children. Then, there are d^i nodes at distance i from v, for all $i \leq n$. So v gets a reward of $a^i b d^i \geq b$ from nodes at distance i from v, for each $i \leq n$. Thus, $R^g(v) \geq nb$. For any $m \in \mathbb{R}^+$, setting $n = m/b$ yields a tree of maximum degree d such that $R^g(v) \geq m$.

Vulnerability to sybil attacks. Geometric reward mechanisms are vulnerable to sybil attacks. Consider node v with n children and no grandchildren. Then, $P(v) = n(ab) - \pi$. Node v could perform a sybil attack, creating the false identities v_1 and v_2. Then, v_2 could buy the product through a direct referral from v_1 and give direct referrals to all of v's children. In this case, $P(v_2) = nab - \pi = P(v)$, and $P(v_1) = na^2 b + ab - \pi$. Thus, $P(v_1) + P(v_2) = P(v) + na^2 b + ab - \pi$. Therefore, by creating false identities v_1 and v_2, v increases its profit if n is sufficiently large.

If v has no ancestors, this sybil attack also results in the seller having to pay an additional $na^2 + ab$ in rewards, while only selling a single extra unit of product. Thus, if it is profitable for v to perform a sybil attack, this attack also reduces the profit of the seller.

The possibility of sybil attacks may also result in buyers referring less influential nodes than they would absent sybil attacks. Suppose v is a child of w. Further suppose that w is only willing to make one referral, and can either refer v or some other node u. If referred, u would make only $n/2$ referrals, so the seller prefers that w refers v instead of u. Without sybil attacks, w would get a reward of $a^2 bn + ab$ for referring v but only $a^2 bn/2 + ab$ for referring u, so w also prefers to refer v. However, if w anticipates that v will perform a sybil attack, w expects to get a reward of only $a^3 bn + a^2 b + ab$ for referring v. For some values of n, a and b this is less than $a^2 bn/2 + ab$, so w will prefer to refer u instead of v, leading to the sale of $n/2$ fewer units.

A node is *profit-maximizing* if it always performs the sybil attack that would most increase its profit, if any. The next lemma shows that if all nodes are profit-maximizing, then in the geometric mechanism each node gets a reward of at most π from each child.

LEMMA 3.1. *In a mechanism that satisfies anonymity and summing contributions, if v's children are profit maximizing then $R(v) \leq |\delta_v^+|\,\pi$.*

PROOF. Let $R(v,u)$ be the total reward v gets from nodes in T_u. Since the mechanism satisfies summing contributions, $R(v) = \sum_{c \in \delta_v^+} R(v,c)$. Suppose all of v's children are profit-maximizing and that $R(v) > |\delta_v^+|\,\pi$. By the pigeonhole principle there exists some $u \in \delta_v^+$ such that $R(v,u) > \pi$. If u replaces itself with two replicas u_1 and u_2 such that u_1 is a child of v, u_2 is the only child of u_1, and $\delta_{u_2}^+ = \delta_u^+$, then, by summing contributions and anonymity, $R(u_2) = R(u)$ and $R(u_1) = R(v,u) > \pi$. Therefore, it would be profitable for u to perform a sybil attack. This contradicts the assumption that u is profit-maximizer. □

Since the geometric reward mechanism satisfies anonymity and summing contributions, Lemma 3.1 implies that if $u \in \delta_v^+$ is a profit-maximizer with regards to sybil attacks, then $R^g(v,u) \leq \pi$. Thus, if buyers are profit maximizers with regards to sybil attacks, the geometric reward mechanism might not satisfy the unbounded reward constraint. Moreover, the next theorem shows that any mechanism that satisfies summing contributions with $c_2 > 0$ is not sybil-proof.

THEOREM 3.2. *There is no sybil-proof mechanism that satisfies summing contributions and gives non-zero rewards for indirect referrals.*

PROOF. We prove the contrapositive. Suppose there is a mechanism that, for some $i > 1$, rewards i-indirect rewards some amount $c_i > 0$. Consider tree T_v with $\lceil \frac{2\pi}{c_i} \rceil$ nodes at distance $i-1$ from v. Node v could perform a sybil attack, creating replica v_1 which refers v_2, and perform all referrals v was making through v_1. Then, the reward of v_2 equals the previous reward of v. On the other hand, v_1 gets a reward of at least $c_i \lceil \frac{2\pi}{c_i} \rceil \geq 2\pi$, from the nodes at distance $i-1$ from v, which are now at distance i from v_1. Then, this sybil attack increases the revenue of v by at least 2π, while only costing v an extra π to buy an additional unit of product. Thus, the mechanism is not sybil-proof. □

3.2. Capping reward mechanisms

In this section we describe a method that takes any anonymous reward mechanism for the tree setting that satisfies the budget and monotonicity constraints and summing contributions and modifies this mechanism to make it sybil-proof. The resulting mechanism still satisfies the budget and monotonicity constraints. If the original mechanism satisfies unbounded rewards, so does the modified mechanism. The modified mechanism no longer satisfies summing contributions, but it does satisfy the subtree constraint. In Section 3.3 we explain how this can be extended to the graph setting.

Let R^0 be an anonymous reward mechanism that satisfies the budget and monotonicity constraints and summing contributions. In particular, R^0 is defined by rewards $\{c_k\}_{k=1}^{\infty}$. The *unit budget* C in R^0 is the maximum reward the seller could have to pay for each unit sold. Under summing contributions, this is: $C := \sum_{i=1}^{\infty} c_i$. The mechanism satisfies the budget constraint if and only if $C \leq \pi$. Thus, a purchase *generates* a reward of C that is divided in some way among ancestors of the node. In this way, a subtree T generates a total reward of $|T|C$.

We now introduce a number of concepts for R^0 which we will use to define our mechanism, R^1. Let the *parent reward* of v, $F^0(v,1)$, be the reward the parent of v gets from T_v under R^0. Then, $F^0(v,1) := \sum_{u \in T_v} c_{\text{dist}(v,u)+1}$. The i^{th} *parent reward* of v, $F^0(v,i)$, is the reward the i^{th} ancestor of v gets from T_v:

$$F^0(v,i) := \sum_{u \in T_v} c_{\text{dist}(v,u)+i}. \tag{1}$$

The total reward given to ancestors of v for purchases from nodes in T_v depends on the number of ancestors v has. However, in order to satisfy the subtree constraint, the reward of v cannot depend on how many ancestors v has. Thus, it is useful to introduce the concept of the *ancestor reward* of v, the total reward that all ancestors of v would get under R^0 from nodes in T_v if v had infinitely many ancestors: $A^0(v) := \sum_{i=1}^{\infty} F^0(v,i)$

By Lemma 3.1, if a mechanism allows a node that has a single child to get a reward of more than π, then it is vulnerable to a local split. To avoid this, we cap the reward a node can get through each child. Let the *modified parent reward* of v, $F^1(v,1)$, be the minimum of the parent reward and π. That is, $F^1(v,1) := \min\{\pi, F^0(v,1)\}$. Under F^1, no node can get more than π through each child. The *base reward* of v, $Q(v)$, is the sum of the modified parent rewards of its children: $Q(v) := \sum_{u\in\delta_v^+} F^1(u,1) = \sum_{u\in\delta_v^+} \min\{\pi, \sum_{u\in T_c} c_{\mathsf{dist}(v,u)}\}$.

The modified mechanism defined by Q is indeed sybil-proof. However, it limits the reward a node can get from each child at π. This reduces the incentive to refer influential people: nodes have no incentive to refer people of influence beyond what is necessary to get a reward of π. In particular, since v can have a reward of at most $|\delta_v^+|\pi$, Q does not satisfy unbounded rewards. Therefore, we further modify Q to obtain a mechanism that is still sybil-proof *and* maintains unbounded rewards. We do this by reassigning the rewards ancestors of v get under R^0 due to T_v, but do not get under Q, to T_v instead.

Our mechanism has two parts. First, we cap the reward a node can get through each child at π. Doing this requires capping not just $F^1(v,1)$, but also capping $F^1(v,i)$ for all i. Second, we redistribute what the cap takes from the ancestors of v to T_v. We start by explaining how the cap works, and why capping $F^1(v,i)$ for $i > 1$ is also necessary.

Example 1. Suppose v has one child u, and u has $\frac{2\pi}{c_2}$ children and no grandchildren. Under R^0, v gets a reward of $c_1 + c_2\frac{2\pi}{c_2} = 2\pi + c_1$. Under Q, v gets only π from nodes in T_u. Let w be the parent of v. Under R^0, w gets $F^0(v,1) = c_1 + F^0(u,2)$ from nodes in T_v. Thus, capping the reward w gets from nodes in T_v at π also requires capping the reward w gets from nodes in T_u at $\pi - c_1$. Similar reasoning requires us to cap $F^1(u,i)$ at $\pi - \sum_{k=1}^{i-1} c_k$.

Formally, the *modified i^{th} parent rewards* of v, $F^1(v,i)$ is the maximum reward the i^{th} ancestor of v can get from T_v under Q. That is, the reward the i^{th} ancestor of v gets from T_v under Q if its only descendants are T_v and ancestors of v. $F^1(v,i)$ is calculated in the same way $F^0(v,i)$ is calculated, but subject to the limitation that each ancestor can get at most π through each of its children. The maximum amount an ancestor of v can get from T_v is π minus the amount it gets from ancestors of v. The amount the i^{th} ancestor of v gets from ancestors of v is c_1 from its child, c_2 from its grandchild, ..., $c_{i-1}b$ from the parent of v. Thus, $F^1(v,i) = \min\{F^0(v,i),\ \pi - \sum_{k=1}^{i-1} c_k\}$. Let

$$K(i) := \pi - \sum_{k=1}^{i-1} c_k, \tag{2}$$

so that $F^1(v,i) = \min\{F^0(v,i),\ K(i)\}$. Note that $K(i) = K(i-1) - c_{i-1}$. Thus, $F^1(v,i) := \min\left\{K(i),\ \sum_{u\in T_v} c_{\mathsf{dist}(v,u)+i}\right\}$.

Now we explain the intuition behind the redistribution.

Example 2. We modify Example 1 so that v has two children, u_1 and u_2, each with $\frac{2\pi}{c_2}$ children and no grandchildren. Each child caps $F^1(u_1,2)$ and $F^1(u_2,2)$ at $\pi - c_1$. Then w, the parent of v, is allocated a reward of $2(\pi - c_1) + c_1$. But the reward w can get is at most π. So there is a surplus of $\pi - c_1$ allocated to w by nodes in T_v that w must reject.

This $\pi - c_1$ is given to v in the redistribution. Thus while the cap limits the reward v gets from each of T_{u_1} and T_{u_2}, the redistribution may give reward to v from nodes in $T_{u_1} \cup T_{u_2}$.

We define the reward given to T_v to be the total reward generated by nodes in T_v (which is $|T_v|C$) minus the reward ancestors of T_v get from T_v. With this definition, a cap on the reward that ancestors of v get from T_v implies a redistribution of the reward: What is taken from ancestors of v is given to nodes in T_v. When calculating the reward of v, some of this extra reward may be taken by descendants of v, any remaining extra reward is taken by v.

Formally, the *modified ancestor reward* of v, $A^1(v)$ is the maximum reward ancestors of v could get from T_v under Q. This is the sum over all i of the $F^1(v,i)$. $A^1(v) := \sum_{i=1}^{\infty} F^1(v,i)$. Then, the reward of tree T_v under R^1 is the reward generated by T_v minus the modified ancestor reward of v $R^1(T_v) := |T_v|C - A^1(v)$. Thus, the reward of an individual node v under R^1 is this amount minus the reward taken by the descendants of v:

$$R^1(v) := R^1(T_v) - R^1(T_v \setminus \{v\}) = |T_v|C - A^1(v) - R^1(T_v \setminus \{v\}).$$

3.3. Extending R^1 to the graph setting

We now show how the concepts we used to define R^1 can be extended to the graph setting. This allows us to modify any mechanism R^0 for the graph setting that satisfies summing contributions, the budget constraint, and monotonicity to obtain a sybil-proof mechanism that satisfies the subtree and budget constraints, and monotonicity.

The i^{th} *contribution* to v of u, $c^i(v,u)$, is the reward the i^{th} ancestor w of v would get from u if $\alpha_w(v) = 1$: that is, if there is a single path p of length i from w to v and all edges in p have weight 1. This is $c^i(v,u) := \sum_{p \in \mathcal{P}_{vu}} c_{|p|+i}\alpha(p)$. If G is a tree with $\alpha_e = 1$ for all $e \in E$, then $|\mathcal{P}_{vu}| = 1$, so $c^i(v,u) = c_{\mathsf{dist}(v,u)+i}$. We generalize the definition of parent reward from the tree setting by replacing $c_{\mathsf{dist}(v,u)+i}$ with $c^i(v,u)$ in (1). Then, in the graph setting the definition of parent reward becomes

$$F^0(v,i) := \sum_{u \in G_v} c^i(v,u) = c_i + \sum_{u \in \delta_v^+} F^0(u, i+1). \tag{3}$$

Similarly, the definition of ancestor reward becomes $A^0(v) := \sum_{i=1}^{\infty} F^0(v,i) = \sum_{i=1}^{\infty} \left[c_i + \sum_{u \in \delta_v^+} F^0(u, i+1)\right]$, by (3).

In the tree setting, we defined $R^1(T_v) := |T_v|C - A^1(v)$. We want to do something similar for the graph setting, but the number of nodes in G_v may not be the relevant value. In particular, if all edges in G_v have very low weight, the reward available for nodes in G_v will be much lower than it would be in an equally sized graph where all edges have weight 1. To account for this, instead of adding C to $R^1(G_v)$ for each node u in G_v, we add only $\alpha_v(u)C$, the reward that is potentially available to v. So,

$$R^1(G_v) := \alpha(G_v)C - A^1(v). \tag{4}$$

Thus, the reward of an individual node v under R^1 is

$$R^1(v) = R^1(G_v) - R^1(G_v \setminus \{v\}) = \alpha(G_v)C - A^1(v) - R^1(G_v \setminus \{v\}). \tag{5}$$

It remains to show how to extend the definition of $A^1(v)$ for the graph setting, for which we need to first extend the definition of $F^1(v,i)$. In the tree setting, $F^1(v,i) = \min\left\{K(i),\ \sum_{u \in T_v} c_{\mathsf{dist}(v,u)+i}\right\}$, which equals $\min\left\{K(i),\ c_i + \sum_{u \in \delta_v^+} F^0(v, i+1)\right\}$. In the tree setting, for all v, i, $F^0(v,i)$ is either greater than $K(i)$ or equal to

$F^1(v,i)$, so we can replace $F^0(v,i)$ with $F^1(v,i)$ in the above expression to get $F^1(v,i) = \min\left\{K(i),\ c_i + \sum_{u\in\delta_v^+} F^1(v,i+1)\right\}$. We add weights to this last expression to obtain our definition of $F^1(v,i)$ for the graph setting: $F^1(v,i) := \min\left\{K(i),\ c_i + \sum_{u\in\delta_v^+} \alpha_{vu}F^1(u,i+1)\right\}$. Finally,

$$A^1(v) := \sum_{i=1}^{\infty} F^1(v,i) = \sum_{i=1}^{\infty} \min\left\{K(i),\ c_i + \sum_{u\in\delta_v^+} \alpha_{vu}F^1(u,i+1)\right\}. \tag{6}$$

3.4. Properties of R^1

All the definitions for the graph setting reduce to the definitions provided for the tree setting in the case in which the referral graph happens to be a tree and all edges have weight 1. Therefore, showing that a property holds for the graph setting implies it also holds for the tree setting.

For all i, $K(i)$ is a constant (see (2)). Then, since $F^0(v,i)$ depends only on the subtree rooted at v, $F^1(v,i)$ also depends only on T_v. This implies that $A^1(v)$ depends solely on T_v. Thus, (5) and (6) show that R^1 satisfies the subgraph constraint. By definition (4), it satisfies the budget constraint: for all $u \in G_v$, $\alpha_v(u) \leq 1$, so $\alpha(G_v) \leq |G_v|$. We now show it also satisfies the monotonicity and unbounded rewards constraints. Lemma 3.3 gives an equivalent way of expressing R^1, which allows us to show that R^1 satisfies monotonicity and unbounded rewards. All missing proofs are contained in the full version of the paper.

LEMMA 3.3. $R^1(v) = \sum_{u\in\delta_v^+} \alpha_{vu}F^1(u,1) + \sum_{i=2}^{\infty} \max\left\{\Big(\sum_{u\in\delta_v^+} \alpha_{vu}F^1(u,i)\Big) - K(i),\ 0\right\}$.

COROLLARY 3.4. *Adding an edge from any node $w \in G_v$ to any node l does not decrease the reward v gets under R^1.*

Proof Sketch: Adding an edge (w,l) creates more paths in the graph without reducing the weights of any path. Then, for any nodes v and u and for all i, adding (w,l) does not decrease $c^i(v,u)$. Thus, it does not decrease $F^1(u,i)$ for any node u or any i. In particular, for any node $u \in \delta_v^+$, for all i, $F^1(u,i)$ does not decrease. Thus, by Lemma 3.3, the reward of v does not decrease. □

By the subgraph constraint, adding an isolated node to a graph G does not change the rewards of any of the nodes in G. Thus, Corollary 3.4 implies that adding a new node u to G and setting some nodes in G as parents of u does not decrease the rewards of any node in G.

In the rest of this section we present the proofs for the tree setting. Proofs for the graph setting proceed similarly.

F^0 is *unbounded* if there exists an integer d such that, for any number $m \in \mathbb{R}^+$ and for any integer i, there exists a subgraph G_v of maximum out-degree d such that $F^0(v,i) \geq m$.

LEMMA 3.5. *If F^0 is unbounded, R^1 satisfies unbounded rewards.*

PROOF. If F^0 is unbounded, there exists an integer d such that, for any integer i, there exists a tree T_v such that $F^0(v,i) \geq \pi$. Then, in T_v, $F^1(v,i) = \pi - \sum_{k=1}^{i-1} c_k$, by definition of F^1. By monotonicity of R^0 and definition of F^0, for any $j \leq i$, $F^0(v,j) \geq$

$F^0(v,i)$. Thus, for every $j \le i$, $F^0(v,j) \ge \pi$, so

$$F^1(v,j) = \pi - \sum_{k=1}^{j-1} c_k. \tag{7}$$

We construct a tree with root w, which has two children, v_1 and v_2. Each of v_1 and v_2 is the root of a tree identical to T_v. Since w has two children, and T_v has maximum degree d, this tree has maximum degree $\max\{d,2\}$. Then, for any $2 \le j \le i$, $\left(\sum_{u\in\delta_w^+} F^1(u,j)\right) - \left(\pi - \sum_{k=1}^{j-1} c_k\right) = F^1(v_1,j) + F^1(v_2,j) - \left(\pi - \sum_{k=1}^{j-1} c_k\right) = 2F^1(v,j) - \left(\pi - \sum_{k=1}^{j-1} c_k\right) = \pi - \sum_{k=1}^{i-1} c$, by (7). So $\left(\sum_{u\in\delta_w^+} F^1(u,j)\right) - (\pi - \sum_{k=1}^{j-1} c_k) \ge \pi - C$. Then, by Lemma 3.3, $R^1(w) \ge \sum_{j=2}^{\infty} \max\left\{\left(\sum_{u\in\delta_v^+} F^1(u,j)\right) - \left(\pi - \sum_{k=1}^{j-1} c_k\right), 0\right\} \ge \sum_{j=2}^{i}(\pi - C) = (i-1)(\pi - C)$. Thus, for any arbitrary $m \in \mathbb{R}^+$, by setting $i \ge \frac{m}{\pi - C} + 1$ we can obtain a tree rooted at w of maximum degree $\max\{d,2\}$ such that $R^1(w) \ge m$. Thus, R^1 satisfies unbounded rewards. □

THEOREM 3.6. *If R^0 satisfies unbounded rewards, then R^1 satisfies unbounded rewards.*

PROOF. We show, by proving the contrapositive, that if R^0 satisfies unbounded rewards, then F^0 is unbounded. By Lemma 3.5, this implies that R^1 satisfies unbounded rewards. Suppose that F^0 is not unbounded. Then, for any integer d, there exists some integers K, i such that for any tree T_u of maximum degree d, $F^0(u,i) \le K$. Then, if $D(v,i) = \{u \in T_v : \text{dist}(v,u) = i\}$ and $D'(v,i) = \{u \in T_v : \text{dist}(v,u) < i\}$, $R^0(v) = \sum_{u\in T_v} c_{\text{dist}(v,u)} = \sum_{u\in D'(v,i)} c_{\text{dist}(v,u)} + \sum_{u\in D(v,i)} F^0(u,i) \le \sum_{u\in D'(v,i)} c_1 + \sum_{u\in D(v,i)} K \le \sum_{l=1}^{i-1} d^l c_1 + d^i K$. Thus, since i, d, c_1 and K are all constants, R^0 is bounded, which completes the proof. □

THEOREM 3.7. *R^1 satisfies monotonicity.*

PROOF. Let T_v be a referral tree with nodes $u \in T_v$, $w \in T_u$. Consider adding a new node y as a child of either u or w. R^1 satisfies monotonicity if and only if the reward of v is at least as high when u is the parent of y as when w is the parent of y. Thus, we compare the change to v's reward in both cases, and show that v's reward increases at least as much if u is the parent of y.

Consider a node $x \in \delta_v^+$. If x is not an ancestor of u or u, then y does not contribute any reward to x whether it is added as a child of u or w. Thus, in this case $F(x,i)$ does not change, for all i. If x is an ancestor of u (or u), then dist(x,y) is greater if y is a child of u than if y is a child of w, since w is a descendant of u. The distance from x to any other node is the same regardless of y's parent. Thus, $c_{\text{dist}(x,y)}$ is greater if y is a child of u than if it is a child of w. Then, for all i, $F(x,i)$ increases more if y is a child of u than if it is a child of w. So, for all $x \in \delta_v^+$, for all i, $F(x,i)$ increases at least as much if y is a child of u as it does if y is a child of w. Then, by Lemma 3.3, $R^1(v)$ increases at least as much if y is a child of u instead of a child of w. Thus, R^1 satisfies monotonicity. □

4. SYBIL ATTACKS

In the absence of sybil attacks, each individual is the owner of at most one node. Let $G = (V,E)$ be a referral graph, and let U be the set of children of node $w \in V$. $G' = (V',E')$ is obtained from G by a *split* at w if:

(1) $V' = V - \{w\} + \{w_1, \ldots, w_l\}$, $l > 1$. The set of nodes $W = w_1, \ldots, w_l$ is the set of *replicas* of w.

(2) All replicas of w have at least one child.
(3) For all $u \in U$, $G_u = G'_u$.
(4) The parents of a replica of w are a subset of the following: other replicas of w, the parents of w in G, and null.
(5) $\forall u \in U$, $\exists$ a replica w_j of w s.t. $u \in \delta^+_{w_j}$.

Note that a sybil attack at w leaves $G - G_w$ unchanged.

A split of G at v is *optimal* if there is no split of G at v that is at least as profitable and creates fewer replicas of v. Then, if it is possible to perform a profitable split, it is possible to perform an optimal split. A *root replica* of v is a replica of v which does not have a replica of v as an ancestor.

LEMMA 4.1. *Let G' be a graph obtained from graph G by a split at v and which contains multiple root replicas of v. Then, there exists a graph G'' which can be obtained from G by a split at v such that G'' is at least as profitable for v under R^1 as G', and G'' contains the same number of replicas of v as G' but only one root replica.*

PROOF. Let G be a referral graph. Suppose a split at v produces graph G', which contains root replicas v_1, v_2. Consider replacing all incoming edges to v_2 (if any) by an edge from v_1 to v_2, with $\alpha_{v_1 v_2} = 1$. Then, by the subgraph constraint, the reward of v_2 does not change, and the reward of replicas other than v_1 does not change. By Corollary 3.4, the reward of v_1 does not decrease. Thus, the reward of v does not decrease. □

LEMMA 4.2. *Let G' be obtained from referral graph G by a split at v such that G' has at least one non-root replica of v with non-replicas as parents. Then, there exists a graph G'' which can be obtained from G by a split at v that is at least as profitable for v under R^1 as G', and which satisfies the following: G'' contains as many replicas of v as G' does; and all non-root replicas of v in G'' have only replicas of v as parents.*

PROOF. Let G be a referral graph. Suppose a split at v produces graph G', which contains root replica v_1 and non-root replica v_2. Suppose $p \in \delta^-_{v_2}$ is not a replica of v. Then, consider replacing the edge from p to v_2 by an edge of equal weight from v_1 to v_2. This is equivalent to deleting the edge from p to v_2 and then adding an edge with the same weight from v_1 to v_2. By definition of split, p is not a descendant of any replica of v. Thus, by the subgraph constraint, eliminating the edge from p to v_2 does not change the reward of any replica of v. By Corollary 3.4, adding an edge from v_1 to v_2 does not reduce the rewards of any node in G_{v_1}, and by the subgraph constraint does not change the rewards of any nodes not in G_{v_1}. Thus, replacing the edge from p to v_2 with an edge of equal weight from v_1 to v_2 does not reduce the reward of any replica of v. □

LEMMA 4.3. *Let G' be obtained from referral graph G by a split at v such that G' contains a single root replica of v, and that replica has multiple children. Then, there exists a graph G'' which can be obtained from G by a split at v such that: it is at least as profitable for v under R^1 as G'; G'' contains as many replicas of v as G'; there is a single root replica of v and that replica has a single child.*

PROOF. Let G be a referral graph. Suppose a split at v produces graph G', containing root replica v_1 with multiple children.

Lemma 4.2 proves that a profit-maximizing sybil attack performed by v results in a single root replica v_1 of v, and that all other replicas of v have only other replicas of v as parents. Since referral graphs do not contain cycles, there must exist a replica v_2 of v that has v_1 as its single parent (the second replica of v from the topologically sorted nodes in the graph). Since v_2 has v_1 as a single parent, the weight of the edge from v_1 to v_2 is 1. Then, by Lemma 4.2, there exists a replica v_2 of v that is a child of v_1, with

$\alpha_{v_1 v_2} = 1$. Then,

$$\begin{aligned} R^1(\{v_1, v_2\}) &= R^1(v_1) + R^1(v_2) \\ &= \alpha(G_{v_1})C - R^1(G_{v_1} \setminus \{v_1\}) - A^1(v_1) + R^1(v_2) \\ &= \alpha(G_{v_1})C - R^1(G_{v_1} \setminus \{v_1, v_2\}) - A^1(v_1). \end{aligned} \tag{8}$$

Suppose v_1 has a child $u \neq v_2$, which may or may not be a replica of v (Figure 1). Now,

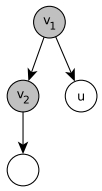

Fig. 1.

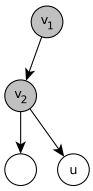

Fig. 2.

consider setting the source of the edge from v_1 to u to be v_2 instead, so that u is no longer a child of v_1 (Figure 2). By the subgraph constraint, this does not change the rewards of nodes in G_{v_1} other than v_1, v_2, nor does it change $\alpha(G_{v_1})$. Thus, by (8), the change in $R^1(\{v_1, v_2\})$ is equal to the change in $-A^1(v_1)$.

Consider any $x \in G_u$. For each path p from v_1 to x going through u, an additional edge has to be appended to the beginning of the path (the edge from v_1 to v_2). This increases the length of the path, reducing $c_{|p|}$, while not increasing the weight of the path. Thus, $c_{|p|}\alpha(p)$ decreases, while neither the length nor the weight of paths from v_1 to x not going through u changes. Thus, for all i, $c^i(v_1, x)$ decreases, so $F^1(v_1, i)$ does not increase. Thus, $A^1(v_1)$ does not increase, so $-A^1(v_1)$ did not decrease, so v's profit did not decrease. □

LEMMA 4.4. *Under R^1, if a split results in a graph where a root replica has a single child, it is not optimal.*

PROOF. Let G be a referral graph. Suppose a split at v produces graph G', which contains root replica v' that has a single child u. Then,

$$\begin{aligned} R^1(v') &= \alpha_{v'u} F^1(u, 1) + \sum_{i=2}^{\infty} \max\{\alpha_{v'u} F^1(u, i) - K(i), 0\}, \qquad \text{by Lemma 3.3.} \\ &\leq F^1(u, 1) + \sum_{i=2}^{\infty} \max\{F^1(u, i) - K(i), 0\}. \end{aligned}$$

By definition, $F^1(u, i) \leq K(i)$, for all i. Thus, $F^1(u, 1) \leq \pi$, and, for all $i \geq 2$, $F^1(u, i) - K(i) \leq 0$. Then, $R^1(v') \leq \pi$. Thus, removing node v' from the graph and setting $\delta_u^- = \delta_{v'}^-$ does not decrease v's profit. Therefore, removing v' yields a split that is as profitable as the original but contains one less replica of v. Thus, the original split is not optimal. □

THEOREM 4.5. *R^1 is sybil-proof.*

PROOF. By definition of split, a split at v must result in at least one root replica of v. If a split results in more than one root replica, then by (4), there exists a split with a single root replica of v that is at least as profitable for v. We can then, without loss of generality, consider only splits that result in a single root replica of v. But that single root replica of v either has multiple children or a single child. If it has multiple

children, there exists a split that results in a single root replica that has a single child, which contains the same number of replicas of v and that is at least as profitable for v. But, if the single root replica of v has a single child, then by Lemma 4.4, it is not optimal. Then, no optimal split can exists, so R^1 is sybil-proof. □

5. COLLUSION

We now consider *collusion sybil attacks*: sybil attacks which are the result of coordinated action by a group of agents $A = \{v_1, \ldots, v_k\}$. Given a graph G, each of this agents may perform a split at their node in the graph, so the resulting graph is the result of multiple splits at multiple points in the graph, with replicas $A' = \{v'_1, \ldots, v'_l\}$ of the original agents (with $l \geq k$).

Agents are coordinating with each other, so they can strategically decide which node to buy from. They may cede referrals to other members of the group. Suppose v is the first node of this group to purchase product. Then, the group could perform a split at v, with some of the resulting replicas being purchases by other members of the group (this includes the possibility of some members of the group buying directly from the seller). Other members of the group could wait and purchase from a descendant of v that is not a replica, creating additional replicas at that point in the graph.

Since agents in the group can transfer referrals to each other, we can see this multiple splits process as starting with the initial node v, and all children of a member of the group being children of v. Let G_v be the this initial tree, before any split is perform or any additional member of the group buys any product. Then

(1) Each member of the group purchases a unit of product somewhere in the graph.
(2) The children of v are divided among the members of the group.
(3) A split is performed at each $v' \in A$.

We want to analyze how the resulting graph differs from a split at v when we consider all members of the group replicas of v. By checking each point of our definition of split in Section 4, we can see that the only points that may not be satisfied when performing multiple splits in this way are points 4, since the parent of a replica can additionally be a descendant of v, and point 2, since some member of the group may end up with no referrals.

We introduce a relaxed definition of split that allows for these differences. Let a *collusion split* be a split as defined in Section 4 but with points 4 and 2 relaxed: the parent of a replica of v is additionally allowed to be any of the descendants of v, and replicas of v may have no children. For clarity, we will now refer to a split as defined in Section 4 as a *simple split*.

Then, the case where sybil attacks are the result of coordinated action by a group of agents differs from the simple case in the following two ways:

— Splits may be collusion splits in addition to strictly simple splits.
— Each individual agent has utility for a unit of product, so a group of size k wants to buy at least $k \geq 1$ units of product (and will only buy more than k if doing so increases its profit). Therefore, a group of size k will perform a collusion split that results in at least k copies.

A reward mechanism is *collusion proof* if for every collusion split that results in $m > k$ replicas of v there exists a collusion split that results in exactly k replicas of v which results in at least as much profit for v. That is, there is no incentive for a group to buy more units of product than they are interested in owning.

THEOREM 5.1. *R^1 is collusion proof.*

6. A SPECIFIC MECHANISM AND IMPLEMENTATION

One concrete anonymous mechanism that satisfies summing contributions, the budget constraint, monotonicity and unbounded rewards is the geometric mechanism described in Section 3.1. Modifying this mechanism in the way described in Section 3.2 yields an anonymous, sybil-proof mechanism that satisfies monotonicity, unbounded rewards, and the subtree and budget constraints. In this section we show that this modified version of the geometric mechanism for the tree setting is simple to implement: updating the rewards of nodes on a referral tree T after a new node u is added requires time linear in the length of the path from the root of T to u.

Recall that the geometric mechanism is defined by two parameters a, b, and that $R^0(v) = \sum_{u \in T_v \setminus \{v\}} a^{\text{dist}(v,u)} b$. Then, $F^0(v,i) = a^i(R^0(v)+b)$ and $F^1(v,i) = \min\{a^i(R^0(v)+b), K(i)\}$, where $c_k = a^k b$. So $C = \sum_{k=1}^{\infty} a^k b = \frac{ab}{1-a}$.

At each node $v \in T$ we maintain values $R^0(v)$, $R^1(v)$, $|T_v|$, $R^1(T_v \setminus \{v\})$, and $A^1(v)$. When v is first added as a leaf to T, $R^0(v)$, $R^1(v)$, and $R^1(T_v \setminus \{v\})$ are initialized to 0, $|T_v|$ is initialized to 1, and $A^1(v)$ is initialized to C. Then, when a leaf u is added to T_v, we execute UPDATE (parent$(u), 1, u$). UPDATE takes three parameters: the node v to update, the distance from v to the newly added node u, and the child c of v such that $u \in T_c$. It updates the values of $R^0(v)$, $|T_v|$, $A^1(v)$, calls UPDATE to update parent(v), and then updates $R^1(T_v \setminus \{v\})$ and $R^1(v)$.

Algorithm 1 UPDATE

Input: $v, \text{dist}(u,v), c$

1: $|T_v| \leftarrow |T_v| + 1$
2: $R^0(v) \leftarrow R^0(v) + c_{\text{dist}(u,v)}$
3: $A^1(v) \leftarrow$ COMPUTE$A^1(R^0(v))$ /* computes the new value of $A^1(v)$ */
4: **if** parent$(v) \neq$ null **then**
5: Update(parent(v), dist$(u,v)+1$, v)
6: **end if**
7: $R^1(T_v \setminus \{v\}) \leftarrow R^1(T_v \setminus \{v\}) - R^1(T_c \setminus \{c\}) - R^1(c) + |T_c|C - A^1(c)$
8: $R^1(v) \leftarrow |T_v|C - A^1(v) - R^1(T_v \setminus \{v\})$

In line 7, updating $R^1(T_v \setminus \{v\})$ requires increasing its value by the difference between the value of $R^1(T_c)$ before u is added and the value of $R^1(T_c)$ after u is added. So, we subtract from $R^1(T_v \setminus \{v\})$ the old value of $R^1(T_c)$, which is $R^1(T_c \setminus \{c\}) + R^1(c)$, since neither $R^1(T_c \setminus \{c\})$ nor $R^1(c)$ were updated yet (those are the only two values that are updated after the call to update the parent). Then, we add the new value of $R^1(T_c)$, which is $|T_c|C - A^1(v)$ by (4), because both $|T_c|$ and $A^1(c)$ have been updated already.

UPDATE is executed once for each ancestor of u, so it is executed at most height of T times. Trivially, everything in this algorithm except possibly the call to COMPUTEA^1, which updates $A^1(v)$, takes constant time. We now examine COMPUTEA^1 to show that it also requires constant time. By definition of A^1, $A^1(v) = \sum_{i=1}^{\infty} F^1(v,i)$. COMPUTE$A^1$ breaks this infinite sum into three parts, and computes each one separately. We show that each of these computations can be done in constant time, and so $A^1(v)$ can be computed in constant time. The first sum is over all indices j for which $F^0(v,j) \geq \pi$. That is, over all indices j such that $a^j(R^0(v)+b) \geq \pi$. Solving for j we get that $j \leq \log_a \frac{\pi}{R^0(v)+b}$. Thus, for $i_1 := \lfloor \log_a \frac{\pi}{R^0(v)+b} \rfloor$, this first sum is $\sum_{j=1}^{i_1} F^1(v,j)$. The second sum is over all indices j for which $F^0(v,j) \leq \pi - C$. That is, over all indices j such that $a^j(R^0(v)+b) \leq \pi - C$. Solving for j we get that $j \geq \log_a \frac{\pi - C}{R^0(v)+b}$. Thus,

for $i_2 := \lceil \log_a \frac{\pi - C}{R^0(v)+b} \rceil$, the second sum $\sum_{j=i_2}^{\infty} F^1(v,j)$. This leaves the third sum as $\sum_{j=i_1+1}^{i_2-1} F^1(v,j)$. Summarizing,

$$A^1(v) = \sum_{j=1}^{i_1} F^1(v,j) + \sum_{j=i_1+1}^{i_2-1} F^1(v,j) + \sum_{j=i_2}^{\infty} F^1(v,j).$$

By definition of $K(i)$, $F^1(v,j) = \min\{F^0(v,j), K(j)\}$. Note that for all j, $\pi - C < K(j) \leq \pi$.

To compute $\sum_{j=i_2}^{\infty} F^1(v,j)$, we use the fact that for all $j \geq i_2$, $F^0(v,j) \leq \pi - C \leq K(j)$. Thus $F^1(v,j) = F^0(v,j) = a^j(R^0(v)+b)$. Therefore, $\sum_{j=i_2}^{\infty} F^1(v,j) = \sum_{j=i_2}^{\infty} a^j(R^0(v)+b) = \frac{a^{i_2}(R^0(v)+b)}{(1-a)}$.

To compute $\sum_{j=1}^{i_1} F^1(v,j)$ we use the fact that for all $j \leq i_1$, $F^0(v,j) \geq \pi \geq K(j)$, so $F^1(v,j) = K(j)$. Let $K_1(i) := \sum_{j=1}^{i} K(j)$. Thus, $\sum_{j=1}^{i_1} F^1(v,j) = \sum_{j=1}^{i_1} K(j) = K_1(i_1)$. It remains to show that we can calculate $K_1(i_1)$ in constant time.

To compute $\sum_{j=i_1+1}^{i_2-1} F^1(v,i)$ we compute, for every $i_1 + 1 \leq j \leq i_2 - 1$, both $F^0(v,j)$ and $K(j)$, and sum $\min\{F^0(v,j), K(j)\}$ over all j. There are at most $i_2 - i_1 - 1 \leq (\log_a \frac{\pi - C}{R^0(v)+b} + 1) - \log_a \frac{\pi}{(R^0(v)+b)} - 1 = \log_a \frac{\pi - C}{\pi}$ such elements to sum over. For every j, $F^0(v,j) = a^j R^0(v)$, so it remains to show that we can calculate each $K(j)$ in constant time.

We now show that we can calculate $K_1(i_1)$ and $K(j)$ for every $i_1 + 1 \leq j \leq i_2 - 1$ in constant time. For all $j > 1$, $K(j) = K(j-1) - c_{j-1}$ and $K_1(j) = K_1(j-1) + K(j)$. $K(1) = K_1(1) = \pi$. Then, $K(i_1)$ can be computed recursively. If we store all K values computed while computing $K(i_1)$, we can then compute $K_1(i_1)$ recursively. We maintain a single table, shared by all nodes in the tree, with computed K, K_1 values. We initialize $K(1) = K_1(1) = \pi$.

Algorithm 2 COMPUTEA^1

Input: $R^0(v)$
Output: $A^1(v)$

1: $i_2 \leftarrow \lceil \log_a \frac{\pi - C}{R^0(v)} \rceil$; $i_1 \leftarrow \lfloor \log_a \frac{\pi}{R^0(v)} \rfloor$
2: $A^1(v) \leftarrow \frac{a^{i_2} R^0(v)}{(1-a)}$ /* sum from $i_2 \rightarrow \infty$. */
3: $K(i_1) \leftarrow K(i_1 - 1) + c_{i_1 - 1}$
4: $K_1(i_1) \leftarrow K(i_1 - 1) + K(i_1)$
5: $A^1(v) \leftarrow A^1(v) + K_1(i_1)$ /* sum from $1 \rightarrow i_1$. */
6: **for** $j = i_1 + 1 \rightarrow i_2 - 1$ **do** /* sum from $i_1 + 1 \rightarrow i_2 - 1$. */
7: $A^0(v) \leftarrow a^i(R^0(v) + b)$
8: $K(j) \leftarrow K(j-1) + c_{j-1}$
9: $A^1(v) \leftarrow A^1(v) + \min\{A^0(v), K(j)\}$
10: **end for**

We only need to show that when computing $K_1(i_1)$ we already have $K_1(i_1 - 1)$ and $K(i_1 - 1)$ and that when we compute $K(j)$ during the loop at the end of COMPUTEA^1 we already have $K(j-1)$.

LEMMA 6.1. *Let T_v be a tree to which we add a leaf u. Let i_1 be the smallest integer j for which $F^0(v,j) \geq \pi$ after adding u. Then, $K(i_1 - 1)$ and $K_1(i_1 - 1)$ are computed before u is added to T_v.*

Thus, $K(i_1 - 1)$ and $K_1(i_1 - 1)$ are computed before the call to update v, so $K(i_1)$ and $K_1(i_1)$ can be computed directly.

During the first iteration of the loop at the end of COMPUTEA^1, $j = i_1+1$, so $j-1 = i_1$. $K(j - 1) = K(i_1)$ was calculated in line 3, so $K(j)$ can be computed in constant time. During successive iterations, $K(j-1)$ was computed during the previous loop iteration, so $K(j)$ can be computed in constant time.

Therefore, updating $A^1(v)$ when a new leaf is added to T_v in a bottom-up manner requires constant time under a modified geometric mechanism. Thus, updating the rewards of a node v after a leaf is added to T_v requires constant time. Since, by the subtree constraint, when adding a new leaf u only the rewards of the ancestors of u can change, updating the rewards of all nodes in a tree when a new leaf is added requires linear time on the length of the path from the root of T to u.

REFERENCES

2009. DARPA network challenge. http://balloon.media.mit.edu/mit/rules.

ABBASSI, Z. AND MISRA, V. 2011. Multi-level revenue sharing for viral marketing. In *Proceedings of ACM NetEcon 2011*.

ARTHUR, D., MOTWANI, R., SHARMA, A., AND XU, Y. 2009. Pricing strategies for viral marketing on social networks. In *Proceedings of the 5th International Workshop on Internet and Network Economics*. WINE '09. Springer-Verlag, Berlin, Heidelberg, 101–112.

BABAIOFF, M., DOBZINSKI, S., OREN, S., AND ZOHAR, A. 2011. On bitcoin and red balloons.

CEBRIAN, M., COVIELLO, L., VATTANI, A., AND VOULGARIS, P. 2011. Query incentive networks with split contracts: Robustness to individuals' selfishness.

DOUCEUR, J. R. AND MOSCIBRODA, T. 2007. Lottery trees: Motivational deployment of networked systems. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*. SIGCOMM '07. ACM, New York, NY, USA, 121–132.

DRUCKER, F. AND FLEISCHER, L. 2012. Simple sybil-proof mechanisms for multi-level marketing. www.cs.dartmouth.edu/ druckerf/papers/sybil-abstract.pdf.

EMEK, Y., KARIDI, R., TENNENHOLTZ, M., AND ZOHAR, A. 2011. Mechanisms for multi-level marketing. In *Proceedings of the 12th ACM Conference on Electronic Commerce*. New York, NY, USA, 209–218.

GRANOVETTER, M. 1978. Threshold models of collective behavior. *The American Journal of Sociology 83,* 6, 1420–1443.

HARTLINE, J., MIRROKNI, V., AND SUNDARARAJAN, M. 2008. Optimal marketing strategies over social networks. In *Proceedings of the 17th international conference on World Wide Web*. WWW '08. ACM, New York, NY, USA, 189–198.

KEMPE, D., KLEINBERG, J., AND TARDOS, E. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '03. ACM, New York, NY, USA, 137–146.

KLEINBERG, J. 2007. Cascading behavior in networks: Algorithmic and economic issues. In *Algorithmic Game Theory*, N. Nisan, T. Roughgarden, va Tardos, and V. V. Vazirani, Eds. Cambridge University Press, Chapter 24, 613–632.

KLEINBERG, J. AND RAGHAVAN, P. 2005. Query incentive networks. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '05. IEEE Computer Society, Washington, DC, USA, 132–141.

LESKOVEC, J., ADAMIC, L. A., AND HUBERMAN, B. A. 2007a. The dynamics of viral marketing. *ACM Trans. Web 1*.

LESKOVEC, J., KRAUSE, A., GUESTRIN, C., FALOUTSOS, C., VANBRIESEN, J., AND GLANCE, N. 2007b. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '07. ACM, New York, NY, USA, 420–429.

PICKARD, G., RAHWAN, I., PAN, W., CEBRIÁN, M., CRANE, R., MADAN, A., AND PENTLAND, A. 2010. Time critical social mobilization: The darpa network challenge winning strategy. *CoRR abs/1008.3172*.

RICHARDSON, M. AND DOMINGOS, P. 2002. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '02. ACM, New York, NY, USA, 61–70.

SINGER, Y. 2012. How to win friends and influence people, truthfully: Influence maximization mechanisms for social networks. In *The ACM Conference on Web Search and Data Mining (WSDM)*.