Chapter 5
Logic Programming Based Diffusion Models

5.1 Introduction

This chapter focuses on a logic-programming approach to social network diffusion first introduced in [23] and later extended in [4]. The advantage with this approach is that we can not only consider the topology of the network, but also consider labeled attributes of the nodes and edges in a natural way. Since its introduction, there have been other variants of the logic-based approach that have leveraged formalisms such as PSL [2] and modal logic [24] in addition to tackling problems such as non-monotonic diffusion reasoning [25] and informing the creation of diffusion-specific centrality measures [26]. These approaches differ from some of the diffusion models in previous chapters in several key ways. For instance, the other models largely assume that a social network is nothing but a set of vertices and edges [16–18]. In contrast, in this chapter we adopt a richer model where edges and vertices can both be labeled with properties. For instance, a political campaigner hoping to spread a positive message about a campaign needs to use demographics (e.g. sex, age group, educational level, group affiliations, etc.) for targeting a political message—a "one size fits all" message will not work. In general, social network researchers would say that they have several sociomatrices that can be used for such applications. Another key difference is that the approaches of the previous chapters reason about a single diffusion model, rather than develop a framework for reasoning about a whole class of diffusion models.

Past diffusion models developed in a variety of fields ranging from business [10], economics [21], social science [20], epidemiology [15, 22, 27], mobile phone usage [11] show that diffusion models vary dramatically from application to application. In this chapter, we organize these models into three broad categories.

1. *Cascade models* [15, 22, 27] are widespread in epidemiology and assume that diffusions are largely based on connectivity between nodes and are largely probabilistic.

2. *Tipping models* do not use probabilities, but use various quantitative calculations to determine when a vertex adopts (or is infected with) a diffusive property. They are omnipresent in the social sciences and business [7, 12, 20]. Nobel-laureate Tom Schelling makes a similar point that diffusions in many social science applications have a tipping point when vertices become influenced by the number of neighbors and the strength of commitment the neighbors may have to a certain position. No probabilities are present in such models.
3. *Homophilic models* are ones where similarity between users, rather than networks effects, dominate diffusion. Similarity is usually calculated using some quantitative model, often related to distance between vectors representing (values of) properties of nodes. For example, [11] tracks adoption of mobile applications in a study of over 27M users and shows that homophily—similarity between users—is the most compelling diffusion model. There are no probabilities here, just similarity measures. Another world famous diffusion model focused on marketing [10] also is based on homophily and similarity of nodes' intrinsic properties rather than a probability.

Moreover, many models use a mix of the above forms. For instance, Cha et al. [5] argues that the way photos are marked as "favorites" on Flickr is based on a mix of cascading and homophilic behavior and to study the former, one must also account for the latter. A similar combination of cascading and tipping is observed in [21]. In general, a language to express diffusion models must be capable of expressing a wide variety of *quantitative* methods encapsulated in the above.

In this chapter, we first show that a class of the well-known generalized annotated program (GAP) paradigm [6] form a convenient method to express many diffusion models. We focus on reasoning with diffusion models (expressed via GAPs) *after the diffusion models have been learned*. In particular, we consider the problem of optimal decision making in social networks which have associated diffusion models expressible as Linear GAPs, though many of the results in this chapter apply to arbitrary GAPs as well. Here are two examples.

- **(Q1) Cell Phone Plans** A cell phone company is promoting a new cell phone plan—as a promotion, it is giving away k free plans to existing customers.[1] Which set of k people should they pick so as to *maximize* the *number* of plan adoptees predicted by a cell phone plan adoption diffusion model they have learned from their past promotions?
- **(Q2) Medication Distribution Plan** A government combating a disease spread by physical contact has limited stocks of free medication to give away. Based on a diffusion model of how the disease spreads (e.g. kids might be more susceptible than adults, those previously inoculated against the disease are safe, etc.), they want to find a set of k people who (jointly) maximally spread the disease when

[1]This framework allows us to add additional constraints—for instance, that plans can only be given to customers satisfying certain conditions, e.g.customers deemed to be "good" according to various business criteria.

infected (so that they can provide immediate treatment to these k people in an attempt to halt the disease's spread).[2] Notice that this query corresponds to only one of many different policies that can be considered to deal with the disease spread scenario, that is, we consider the case where a diffusion model expressing how an infected person can infect other people is available and formulate a query that looks at the maximum spread when k people are infected. Other queries, possibly leading to different answers about who should be treated with medications, are possible.

Both these problems are instances of a class of queries that we call *Social Network Diffusion Optimization Problem* (SNDOP) queries. They differ from other queries studied in logic programming in two fundamental ways: (1) They are specialized to operate on graph data where the graph's vertices and edges are labeled with properties and where the edges can have associated weights, (2) They find *sets of vertices* that optimize complex objective functions that can be specified by the user.

This chapter is organized as follows. In Sect. 5.2, we provide an overview of GAPs (past work), define a social network (SN for short), and explain how GAPs can represent some types of diffusion in SNs. Section 5.3 formally defines different types of social network diffusion optimization problems and provides results on their computational complexity and other properties. Section 5.4 shows how our framework can represent several existing diffusion models for social networks including economics and epidemiology. In Sect. 5.5 we present the exact SNDOP-Mon algorithm to answer SNDOP queries under certain assumptions of monotonicity. We then develop a greedy algorithm GREEDY-SNDOP and show that under certain conditions, it is guaranteed to be an $(\frac{e}{e-1})$ approximation algorithm for SNDOP queries—this is the best possible approximation guarantee. Last, but not least, we describe our prototype implementation and experiments in Sect. 5.5. Specifically, we tested our GREEDY-SNDOP algorithm on a real-world social network data set derived from Wikipedia logs. We show that we solve social network diffusion optimization problems over real data sets in acceptable times.

5.2 Embedding Diffusion Models into Annotated Logic Programs

In this section, we first formalize social networks, then briefly review generalized annotated logic programs (GAPs) [6] and then describe how GAPs can be used to represent concepts related to diffusion in social networks.

[2] Again, this framework allows us to add additional constraints—for instance, that medication can only be given to people satisfying certain conditions, e.g. be over a certain age, or be within a certain age range and not have any conditions that are contra-indicators for the medication in question.

5.2.1 Social Networks Formalization

Throughout this chapter, we assume the existence of two arbitrary but fixed disjoint sets VP, EP of *vertex* and *edge predicate symbols* respectively. Each vertex predicate symbol has arity 1 and each edge predicate symbol has arity 2.

Definition 5.1. A **social network** is a 5-tuple $(\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ where:

1. $\mathbf{V}$ is a finite set whose elements are called *vertices*.
2. $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ is a finite *multi-set* whose elements are called *edges*.
3. $\ell_{vert} : \mathbf{V} \to 2^{\mathsf{VP}}$ is a function, called *vertex labeling function*.
4. $\ell_{edge} : \mathbf{E} \to \mathsf{EP}$ is a function, called *edge labeling function*.[3]
5. $w : \mathbf{E} \to [0, 1]$ is a function, called *weight function*.

We now present a brief example of an SN.

Example 5.1. Let us return to the cell phone example (query **(Q1)**). Figure 5.1 shows a toy social network the cell phone company might use. Here, we might have $\mathsf{VP} = \{$*male*, *female*, *adopter*, *temp_adopter*, *non_adopter*$\}$ denoting the sex and past adoption behavior of each vertex; EP might be the set $\{$*phone*, *email*, *IM*$\}$ denoting the types of interactions between vertices (phone call, email, and instant messaging respectively). The function ℓ_{vert} is shown in Fig. 5.1 by the shape (denoting past adoption status) and shading (male/female). The type of edges (bold for phone, dashed for email, dotted for IM) is used to depict ℓ_{edge}. $w(\langle v_1, v_2 \rangle)$ denotes the percentage of communications of type $\ell_{edge}(\langle v_1, v_2 \rangle)$ initiated by v_1 that were with v_2 (measured either w.r.t. time or bytes).

It is important to note that our definition of social networks is much broader than that used by several researchers [10, 11, 22, 27] who often do not consider either ℓ_{edge} or ℓ_{vert} or edge weights through the function w—it is well-known in marketing that intrinsic properties of vertices (customers, patients) and the nature and strength of the relationships (edges) is critical for decision making in those fields.

Note We assume that SNs satisfy various integrity constraints. In Example 5.1, it is clear that $\ell_{vert}(v)$ should include at most one of *male*,*female* and at most one of *adopter*, *temp_adopter*,*non_adopter*. *We assume the existence of some integrity constraints to ensure this kind of semantic integrity*—they can be written in any reasonable syntax to express ICs—in the rest of this chapter, we assume that social networks have associated ICs and that they satisfy them. In our example, we will assume ICs ensuring that a vertex can be marked with at most one of *male*,*female* and at most one of *adopter*, *temp_adopter*, *non_adopter*.

[3]Each edge $e \in \mathbf{E}$ is labeled by exactly one predicate symbol from EP. However, there can be multiple edges between two vertices labeled with different predicate symbols.

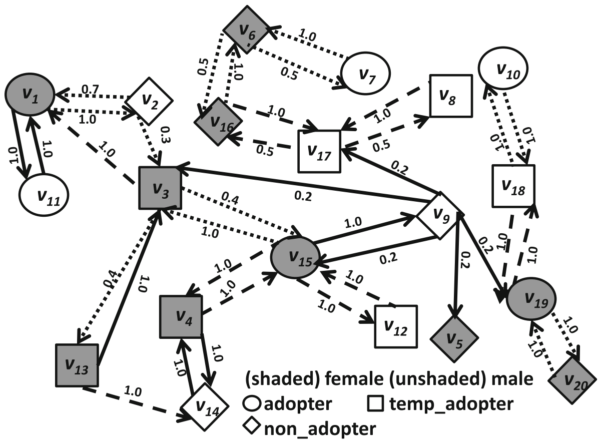

Fig. 5.1 Example cellular network

5.2.2 Generalized Annotated Programs: A Recap

We now recapitulate the definition of generalized annotated logic programs from [6]. We assume the existence of a set AVar of variable symbols ranging over the unit real interval $[0,1]$ and a set $\mathscr{F}$ of function symbols each of which has an associated arity. We start by defining annotations.

Definition 5.2 (Annotation). Annotations are inductively defined as follows:

(i) Any member of $[0,1] \cup \mathsf{AVar}$ is an annotation.
(ii) If $f \in \mathscr{F}$ is an n-ary function symbol and $t_1,\ldots,t_n$ are annotations, then $f(t_1,\ldots,t_n)$ is an annotation.

For instance, $0.5, 1, 0.3$ and X are all annotations (here X is assumed to be a variable in AVar). If $+, *, /$ are all binary function symbols in $\mathscr{F}$, then $\frac{(X+1)*0.5}{0.3}$ is an annotation.[4]

We define a separate logical language whose constants are members of **V** and whose predicate symbols consist of $\mathsf{VP} \cup \mathsf{EP}$. We also assume the existence of a set $\mathscr{V}$ of variable symbols ranging over the constants (vertices). No function symbols are present. Terms and atoms are defined in the usual way (cf. [19]). If $A = p(t_1,\ldots,t_n)$ is an atom and $p \in \mathsf{VP}$ (resp. $p \in \mathsf{EP}$), then A is called a *vertex* (resp. *edge*) atom. We will use $\mathscr{A}$ to denote the set of all ground atoms (i.e., atoms where no variable occurs).

[4]Notice that in [6] annotations are not restricted to be in $[0,1]$ but any upper semi-lattice is allowed—for the purpose of this chapter we will restrict ourselves to the unit real interval.

Definition 5.3 (Annotated Atom/GAP-Rule/GAP). If A is an atom and μ is an annotation, then $A:\mu$ is an *annotated atom*. If A is a vertex (resp. edge) atom, then $A:\mu$ is also called *vertex* (resp. *edge*) annotated atom. If $A_0:\mu_0, A_1:\mu_1,\ldots,A_n:\mu_n$ are annotated atoms, then

$$A_0:\mu_0 \leftarrow A_1:\mu_1 \wedge \ldots \wedge A_n:\mu_n$$

is called a *GAP rule* (or simply *rule*). When $n=0$, the above rule is called a *fact*.[5] A *generalized annotated program* (GAP) is a finite set of rules. An annotated atom (resp. a rule, a GAP) is *ground* iff there are no occurrences of variables from either AVar or $\mathscr{V}$ in it.

Every social network $\mathscr{S}=(\mathbf{V},\mathbf{E},\ell_{vert},\ell_{edge},w)$ can be represented by the GAP $\Pi_{\mathscr{S}}=\{q(v):1\leftarrow \mid v\in\mathbf{V}\wedge q\in\ell_{vert}(v)\}\cup\{ep(v_1,v_2):w(\langle v_1,v_2\rangle)\leftarrow \mid \langle v_1,v_2\rangle\in\mathbf{E}\wedge\ell_{edge}(\langle v_1,v_2\rangle)=ep\}$.

Definition 5.4 (Embedded Social Network). A social network $\mathscr{S}$ is said to be *embedded* in a GAP Π iff $\Pi_{\mathscr{S}}\subseteq\Pi$.

It is clear that all social networks can be represented as GAPs. When we augment $\Pi_{\mathscr{S}}$ with other rules—such as rules describing how certain properties diffuse through the social network, we get a GAP $\Pi\supseteq\Pi_{\mathscr{S}}$ that captures both the structure of the SN and the diffusion principles. Here is a small example of such a GAP.

Example 5.2. The GAP Π_{cell} might consist of $\Pi_{\mathscr{S}}$ using the social network of Fig. 5.1 plus the GAP-rules:

1. $\mathit{will\_adopt}(V_0):0.8\times X+0.2\leftarrow \mathit{adopter}(V_0):1\wedge \mathit{male}(V_0):1\wedge$
   $\mathit{IM}(V_0,V_1):0.3\wedge \mathit{female}(V_1):1\wedge \mathit{will\_adopt}(V_1):X.$
2. $\mathit{will\_adopt}(V_0):0.9\times X+0.1\leftarrow \mathit{adopter}(V_0):1\wedge \mathit{male}(V_0):1\wedge$
   $\mathit{IM}(V_0,V_1):0.3\wedge \mathit{male}(V_1):1\wedge \mathit{will\_adopt}(V_1):X.$
3. $\mathit{will\_adopt}(V_0):1\leftarrow \mathit{temp\_adopter}(V_0):$
   $1\wedge \mathit{male}(V_0):1\wedge \mathit{email}(V_1,V_0):1\wedge \mathit{female}(V_1):1\wedge \mathit{will\_adopt}(V_1):1.$

Rule (1) says that if V_0 is a male adopter and V_1 is female and the weight of V_0's instant messages to V_1 is 0.3 or more, and we previously thought that V_1 would be an adopter with confidence X, then we can infer that V_0 will adopt the new plan with confidence $0.8\times X+0.2$. The other rules may be similarly read.

Suppose $\mathscr{S}$ is a social network and $\Pi\supseteq\Pi_{\mathscr{S}}$ is a GAP. In this case, we call the rules in $\Pi-\Pi_{\mathscr{S}}$ *diffusion rules*. In this chapter we consider a restricted class of GAPs: every rule with a non-empty body has a vertex annotated atom in the head ([6] allows any atom to appear in the head of a rule). Thus, edge atoms can appear only in rule bodies or facts. This means that neither edge weights nor edge labels change as the result of the diffusion. However, for the general case, it is possible for them to change as a result of the diffusion process.

[5]For notational simplicity, we will often write a fact $A_0:\mu_0\leftarrow$ simply as $A_0:\mu_0$, i.e. we drop the symbol $\leftarrow$.

GAPs have a formal semantics that can be immediately used. An interpretation I is any mapping from the set $\mathscr{A}$ of all grounds atoms to $[0,1]$. The set $\mathscr{I}$ of all interpretations can be partially ordered via the ordering: $I_1 \preceq I_2$ iff for all ground atoms A, $I_1(A) \leq I_2(A)$. $\mathscr{I}$ forms a complete lattice under the $\preceq$ ordering.

Definition 5.5 (Satisfaction/Entailment). An interpretation I *satisfies* a ground annotated atom $A : \mu$, denoted $I \models A : \mu$, iff $I(A) \geq \mu$. I satisfies a ground GAP-rule r of the form $AA_0 \leftarrow AA_1 \wedge \ldots \wedge AA_n$ (denoted $I \models r$) iff either (i) I satisfies AA_0 or (ii) there exists an $1 \leq i \leq n$ such that I does not satisfy AA_i. *I satisfies* a non-ground annotated atom (rule) iff I satisfies all ground instances of it. *I satisfies* a GAP iff I satisfies all rules in it. A GAP Π *entails* an annotated atom AA, denoted $\Pi \models AA$, iff every interpretation I that satisfies Π also satisfies AA.

As shown by Kifer and Subrahmanian [6], we can associate a fixpoint operator with any GAP Π that maps interpretations to interpretations.

Definition 5.6. Suppose Π is any GAP and I an interpretation. The mapping $\mathbf{T}_\Pi$ that maps interpretations to interpretations is defined as $\mathbf{T}_\Pi(I)(A) = \mathbf{sup}\{\mu \,|\, A : \mu \leftarrow AA_1 \wedge \ldots \wedge AA_n$ is a ground instance of a rule in Π and for all $1 \leq i \leq n, I \models AA_i\}$.

The results of [6] show that $\mathbf{T}_\Pi$ is monotonic (w.r.t. $\preceq$) and has a least fixpoint $lfp(\mathbf{T}_\Pi)$. Moreover, they show that Π entails $A : \mu$ iff $\mu \leq lfp(\mathbf{T}_\Pi)(A)$ and hence $lfp(\mathbf{T}_\Pi)$ precisely captures the ground atomic logical consequences of Π. They also define the *iteration* of $\mathbf{T}_\Pi$ as follows: $\mathbf{T}_\Pi \uparrow 0$ is the interpretation that assigns 0 to all ground atoms; $\mathbf{T}_\Pi \uparrow (i+1) = \mathbf{T}_\Pi(\mathbf{T}_\Pi \uparrow i)$.

The semantics of GAPs requires that when there are multiple ground instances of GAP-rules with the same head that "fire", the highest annotation in any of these ground rules is "chosen" according to the semantics of GAPs. This might seem restrictive and counter-intuitive to some, but it actually is the source of much power of GAPs. For instance, one school of thought is that when multiple ground rules with the same head "fire", the annotation derived should be the "noisy-or" value derived by combining the values of the annotations in the heads of firing rules. However, this is just one way of combining evidence from multiple sources many other triangular co-norms other than noisy-or can be used and have been used in the literature. However, such T-norms can be expressed in our framework. If we have ground rules $G_1, G_2, \ldots, G_n$, each having the same atom in the head, and we want to combine evidence using a triangular co-norm[6] $\oplus$, and if G_i has the form:

$$A : \mu_i \leftarrow Body_i$$

then we can replace these rules with the rules:

$$A : \oplus(\{\mu_i \,|\, i \in X\}) \leftarrow \bigwedge_{i \in X} Body_i$$

[6]When we apply $\oplus$ to a set $\{x_1, \ldots, x_k\}$, we use $\oplus(\{x_1, \ldots, x_k\})$ as short-hand for $\oplus(x_1, \oplus(\{x_2, \ldots, x_n\}))$ which is well defined as all triangular co-norms are commutative and associative.

for any subset $X \subseteq \{1,\ldots,n\}$. Moreover, as we have already remarked, many real-world diffusion models are non-probabilistic, making assumptions about how annotations should be combined harder to justify. However, the above discussion shows that the GAP framework is capable of expressing such rules. Though there is clearly a cost in terms of difficulty of expressing such methods to combine evidence generated by multiple rules, algorithms already exist and have been implemented [2] to learn GAP-based diffusion rules automatically from social network time series data.

We will show (in Sect. 5.4) that many existing diffusion models for a variety of phenomena can be expressed as a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$ by adding some GAP-rules describing the diffusion process to $\Pi_{\mathscr{S}}$.

5.3 Social Network Diffusion Optimization Problem (SNDOP) Queries

5.3.1 Basic SNDOP Queries

In this section, we develop a formal syntax and semantics for optimization in social networks, taking advantage of the aforementioned embedding of SNs into GAPs. In particular, we formally define SNDOP queries, examples of which have been informally introduced earlier as **(Q1)** and **(Q2)**. We see from queries **(Q1)** and **(Q2)** that a SNDOP query looks for a **set** $\mathbf{V}'$ of vertices and has the following components: (1) an objective function expressed via an aggregate operator, (2) an integer $k > 0$, (3) a set of conditions that each vertex in $\mathbf{V}'$ must satisfy, (4) an "input" atom $g_I(V)$, and (v) an "output" atom $g_O(V)$ (here g_I and g_O are vertex predicate symbols, whereas V is a variable).

Aggregates It is clear that in order to express queries like **(Q1)** and **(Q2)**, we need aggregate operators which are mappings $agg : \mathsf{FM}([0,1]) \to \mathbb{R}^+$ ($\mathbb{R}^+$ is the set of non-negative reals) where $\mathsf{FM}(X)$ denotes the set of all finite multisets that are subsets of X. Relational DB aggregates like SUM,COUNT,AVG,MIN,MAX are all aggregate operators which can take a finite multiset of non-negative reals as input and return a single non-negative real.

Vertex Condition A vertex condition is a set of vertex annotated atoms containing exactly one variable (intuitively, such annotated atoms are conditions that must be jointly satisfied by a vertex). More formally, a vertex condition VC is a set $\{p_1(V) : \mu_1,\ldots,p_n(V) : \mu_n\}$ where each $p_i \in \mathsf{VP}$, $V \in \mathscr{V}$, and each $\mu_i \in [0,1]$. We use $VC[V/v]$ to denote the set of ground annotated atoms obtained from VC by replacing each occurrence of V with v, that is $VC[V/v] = \{p_1(v) : \mu_1,\ldots,p_n(v) : \mu_n\}$. A GAP Π entails $VC[V/v]$, denoted $\Pi \models VC[V/v]$, iff $\Pi \models p_i(v) : \mu_i$ for all $1 \leq i \leq n$.

Thus, in our example, $\{male(V) : 1, adopter(V) : 1\}$ is a vertex condition, but $\{male(V) : 1, email(V,V') : 1\}$ is not. We are now ready to define a SNDOP query.

Definition 5.7 (SNDOP Query). A *SNDOP query* is a 5-tuple $(agg, VC, k, g_I(V), g_O(V))$ where *agg* is an aggregate, *VC* is a vertex condition, $k > 0$ is an integer, and $g_I(V)$, $g_O(V)$ are vertex atoms.

Let us consider again the medication distribution plan example. Suppose we have a diffusion model expressing how a property *healthy* diffuses in a social network w.r.t. a property *immune* (which would hold for a vertex when a medication is given to it). An interesting query to pose would be to determine a set of at most k people such that if these people were immune to the disease, then the number of healthy people would be maximized. Such a query can be expressed with the SNDOP query $(\mathsf{SUM}, \emptyset, k, immune(V), healthy(V))$. Here, the goal is to find a set $\mathbf{V}' \subseteq \mathbf{V}$ of vertices such that $|\mathbf{V}'| \leq k$ and the following is maximized:

$$\mathsf{SUM}\{lfp(\mathbf{T}_{\Pi \cup \{immune(v'):1 \,|\, v' \in \mathbf{V}'\}})(healthy(v)) \,|\, v \in \mathbf{V}\}$$

Here, the SUM is applied to a multiset rather than a set. Note that in the query above $VC = \emptyset$, meaning that the *immune* property can be assigned to any vertex of the SN. However, other queries can be expressed where *VC* imposes restrictions on which vertices can have property *immune*. As an example, $VC = \{adult(V)\}$ would enforce every vertex in $\mathbf{V}'$ to be an adult person.

If we return to our cell phone example, we can set $agg = \mathsf{SUM}$, $VC = \emptyset$, $k = 3$ (for example), $g_I(V) = will\_adopt(V)$, and $g_O(V) = will\_adopt(V)$ (notice that in this case $g_I(V) = g_O(V)$). Here also, the goal is to find a set $\mathbf{V}' \subseteq \mathbf{V}$ of vertices such that $|\mathbf{V}'| \leq 3$ and the following is maximized:

$$\mathsf{SUM}\{lfp(\mathbf{T}_{\Pi \cup \{will\_adopt(v'):1 \,|\, v' \in \mathbf{V}'\}})(will\_adopt(v)) \,|\, v \in \mathbf{V}\}$$

Here, the SUM is applied to a multiset rather than a set. Note that the diffusion model's impact is captured via the $lfp(\mathbf{T}_{\Pi \cup \{will\_adopt(v'):1 \,|\, v' \in \mathbf{V}'\}})(will\_adopt(v))$ expression which, intuitively, tells us the confidence (according to the diffusion model) that each vertex v will be an adopter. If we return to an extended version of our cell phone example and we want to ensure that the vertices in $\mathbf{V}'$ are "good" customers[7] then we merely can set $VC = \{good(V) : 1\}$. This query now asks us to find a set $\mathbf{V}'$ of three or less vertices—all of which are "good" customers of the company C—such that $\mathsf{SUM}\{lfp(\mathbf{T}_{\Pi \cup \{will\_adopt(v'):1 \,|\, v' \in \mathbf{V}'\}})(will\_adopt(v)) \,|\, v \in \mathbf{V}\}$ is maximized.

Our framework also allows the vertex condition *VC* to have annotations other than 1. So in our cell phone example, the company could explicitly exclude anyone whose "opinion" toward the company is negative. If opinion is quantified on a continuous $[0,1]$ scale (such automated systems do exist [1]), then the vertex condition might be restated as $VC = \{good(V) : 1, negative\_opinion\_C(V) : 0.7\}$

[7] We can think of many ways a company may define "good" customers, e.g. those who regularly pay their bills on time, those who buy a lot of services from the company, those who have stayed as customers for a long time, etc. For our example, the specific definition of "good" is not relevant.

which says that the company wants to exclude anyone whose negativity about the company exceeds 0.7 according to an opinion scoring engine such as [1].

Definition 5.8 (Pre-answer/Value). Consider a social network $\mathscr{S} = (\mathsf{V}, \mathsf{E}, \ell_{vert}, \ell_{edge}, w)$ embedded in a GAP Π. A *pre-answer* to the SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ w.r.t. Π is any set $\mathsf{V}' \subseteq \mathsf{V}$ such that:

1. $|\mathsf{V}'| \leq k$, and
2. for all vertices $v'' \in \mathsf{V}'$, $\Pi \cup \{g_I(v') : 1 \mid v' \in \mathsf{V}'\} \models VC[V/v'']$.

We use $\mathsf{pre\_ans}(Q, \Pi)$ to denote the set of all pre-answers to Q w.r.t. Π (whenever Π is clear from the context we simply write $\mathsf{pre\_ans}(Q)$).

The *value* of a pre-answer V' is defined as follows:

$$value(\mathsf{V}') = agg(\{lfp(\mathbf{T}_{\Pi \cup \{g_I(v'):1 \mid v' \in \mathsf{V}'\}})(g_O(v)) \mid v \in \mathsf{V}\})$$

where the aggregate is applied to a multi-set rather than a set. We also note that we can define *value* as a mapping from interpretations to reals based on a SNDOP query. We say $value(I) = agg(\{I(g_O(v)) \mid v \in \mathsf{V}\})$.

If we return to our cell phone example, V' is the set of vertices to which the company is considering giving free plans. $value(\mathsf{V}')$ is computed as follows.

1. Find the least fixpoint of $\mathbf{T}_{\Pi'_{cell}}$ where Π'_{cell} is Π_{cell} expanded with facts of the form $will\_adopt(v') : 1$ for each vertex $v' \in \mathsf{V}'$.
2. For each vertex $v \in \mathsf{V}$ (the entire set of vertices, not just V' now), we now find the confidence assigned by the least fixpoint.
3. Summing up these confidences gives us a measure of the expected number of plan adoptees.

Definition 5.9 (Answer). Suppose a social network $\mathscr{S} = (\mathsf{V}, \mathsf{E}, \ell_{vert}, \ell_{edge}, w)$ is embedded in a GAP Π and $Q = (agg, VC, k, g_I(V), g_O(V))$ is a SNDOP query. A pre-answer V' is an *answer* to the SNDOP query Q w.r.t. Π iff the SNDOP query has no other pre-answer V'' such that $value(\mathsf{V}'') > value(\mathsf{V}')$.[8]

The *answer set* to SNDOP query Q w.r.t. Π, denoted $\mathsf{ans}(Q, \Pi)$, is the set of all answers to Q w.r.t. Π (whenever Π is clear from the context we simply write $\mathsf{ans}(Q)$).

It is important to note that an answer to an SNDOP query is a **set** of vertices that **jointly** maximize the objective function specified. Thus, it is entirely possible that if we set $k = 1$, we could have two answers $\{a_1\}$ and $\{a_2\}$ each of which ties for the highest value. However, $\{a_1, a_2\}$ may *not* be the answer that optimizes the objective function when $k = 2$.

[8]Throughout this chapter, we only treat maximization problems—there is no loss of generality in this because minimizing an objective function f is the same as maximizing $-f$.

Example 5.3. For instance, suppose a_1 and a_2 are brothers with largely the same connections. The sets $\{a_1\}$ and $\{a_2\}$ both have value 100 each and let us say these constitute an answer (looking at one individual only) w.r.t. an objective function, e.g. influencing voters in an election to vote for candidate X. As a_1, a_2 mostly influence the same people, they may jointly be able to get only 110 people to vote for the candidate because of the large overlap in their sphere of influence. However, now consider persons a_3, a_4. Each of them can only influence 90 voters by themselves, but only 10 of these voters "overlap". Thus, they can jointly influence $80+80+10 = 170$ voters to vote for X. It would make more sense (all other things being equal) for the candidate's party to invest in $\{a_3, a_4\}$.

Example 5.4. Consider the GAP Π_{cell} of Example 5.2 with the social network from Fig. 5.1 embedded and the SNDOP query $Q_{cell} = (SUM, \emptyset, 3, will\_adopt, will\_adopt)$. The sets $\mathbf{V}'_1 = \{v_{15}, v_{19}, v_6\}$ and $\mathbf{V}'_2 = \{v_{15}, v_{18}, v_6\}$ are both *pre-answers*. In the case of $\mathbf{V}'_1$, two applications of the $\mathbf{T}_\Pi$ operator yields a fixpoint where the vertex atoms formed with *will_adopt* and vertices in the set $\{v_{15}, v_{19}, v_6, v_{12}, v_{18}, v_7, v_{10}\}$ are annotated with 1. For $\mathbf{V}_2$, only one application of $\mathbf{T}_\Pi$ is required to reach a fixpoint. In the fixpoint, vertex atoms formed with *will_adopt* and vertices in the set $\{v_{15}, v_6, v_{12}, v_{18}, v_7, v_{10}\}$ are annotated with 1. As these are the only vertex atoms formed with *will_adopt* that have a non-zero annotation after reaching the fixed point, we know that $value(\mathbf{V}'_1) = 7$ and $value(\mathbf{V}'_2) = 6$.

5.3.2 Special Cases of SNDOPs

In this section, we examine several special cases of SNDOPs that still allow us to represent a wide variety of diffusion models. Table 5.1 illustrates the special cases discussed in this section while Table 5.2 illustrates various properties we prove (and the assumptions under which those properties are proved).

Special Cases of GAPs First, we present a class of GAPs called *linear* GAPs. Intuitively, a GAP is linear if the annotations in the rule heads are linear functions and the annotations in the body are variables. It is important to note that a wide variety of diffusion models can be represented with GAPs that meet the requirements of this special case. We formally define linear GAPs below.

Table 5.1 Special cases of SNDOPs

Type	Special case	Reference
Special cases of Π	Linear GAP	Definition 5.10
Special cases of *agg*	Monotonic	Definition 5.11
	Positive-linear	Definition 5.12
Special cases of *value*	Normalized	Definition 5.13
	A-priori *VC*	Definition 5.14

Table 5.2 Properties that can be proven given certain assumptions

Property	Assumptions
Monotonicity of *value* (Lemma 5.1)	Monotonicity of *agg*
Multiset $\{\mathsf{V}' \subseteq \mathsf{V} \mid \mathsf{V}'$ *is a pre-answer*$\}$ is a uniform matroid (Lemma 5.2)	A-priori VC
Submodularity of *value* (Theorem 5.1)	Linear GAP
	Positive-linear *agg*
	A-priori VC

Definition 5.10 (Linear GAP). A GAP-rule is *linear* iff it is of the form:

$$H_0 : c_0 + c_1 \cdot X_1 + \cdots + c_n \cdot X_n \leftarrow A_1 : X_1 \wedge \ldots \wedge A_n : X_n$$

where each $c_i \in [0,1]$, $\Sigma_{i=1}^{n} c_i \in [0,1]$, and each X_j is a variable in AVar. A GAP is linear iff each rule in it is linear.

Special Aggregates We define two types of aggregates: *monotonic* aggregates and *positive-linear* aggregates.

To define monotonicity, we first define a partial order $\sqsubseteq$ on multi-sets of numbers as follows: given two multi-sets of numbers X_1 and X_2, we write $X_1 \sqsubseteq X_2$ iff there exists an *injective* mapping $\beta : X_1 \rightarrow X_2$ such that $\forall x_1 \in X_1, x_1 \leq \beta(x_1)$.

Definition 5.11 (Monotonic Aggregate). An aggregate *agg* is **monotonic** iff whenever $X_1 \sqsubseteq X_2$, it is the case that $agg(X_1) \leq agg(X_2)$.

Definition 5.12 (Positive-Linear Aggregate). An aggregate *agg* is **positive-linear** iff it is defined as follows: $agg(X) = c_0 + \Sigma_{x_i \in X}\, c_i \cdot x_i$, where X is a finite multiset and $c_i \geq 0$ for all $i > 0$.

In the previous definition, note that c_0 can be positive, negative, or 0. Thus, we only require that the coefficients associated with the elements of the multi-set be positive—we allow for an additive constant to be negative. One obvious example of a positive-linear aggregate is SUM. Moreover, any positive weighted sum will also meet these requirements.

Proposition 5.1. *If agg is a positive-linear aggregate, then it is a monotonic aggregate.*

Special Cases of the Query We now describe two special cases of the query: *Normalized* and *a-priori VC* SNDOP queries. Intuitively, normalized means that $value(\emptyset) = 0$.

Definition 5.13 (Normalized). An SNDOP query is **Normalized** w.r.t. a given social network $\mathscr{S}$ and a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$ iff $value(\emptyset) = 0$.

Note that the function *value* is *uniquely defined* by a social network, a SNDOP query, and a diffusion model Π and hence the above definition is well defined.

The following result states that if an SNDOP query Q with a positive-linear aggregate is not normalized, then we can always modify it into an "equivalent" SNDOP query Q' (i.e. $\mathsf{ans}(Q) = \mathsf{ans}(Q')$) which is normalized and still maintains a positive-linear aggregate.

Proposition 5.2. *Let $Q = (agg, VC, k, g_I(V), g_O(V))$ be a SNDOP query which is not normalized w.r.t. a social network $\mathscr{S}$ and a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$, and where agg is positive-linear. Let $agg'(X) = agg(X) - value(\emptyset)$. Then, $Q' = (agg', VC, k, g_I(V), g_O(V))$ is a SNDOP query which is normalized w.r.t. $\mathscr{S}$ and Π, $\mathsf{ans}(Q) = \mathsf{ans}(Q')$, and agg' is positive-linear.*

Recall that in order to check if a set of vertices $\mathbf{V}'$ is a pre-answer, we need to check for all vertices $v'' \in \mathbf{V}'$ if $\Pi \cup \{g_I(v') : 1 \mid v' \in \mathbf{V}'\} \models VC[V/v'']$ (see condition (2) of Definition 5.8). Intuitively, a SNDOP query has an *A-Priori VC* (w.r.t. a given social network $\mathscr{S}$ and a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$) when we can check this condition by looking only at the original social network $\mathscr{S}$ (thereby disregarding Π), that is we can check for all vertices $v'' \in \mathbf{V}'$ if $\Pi_{\mathscr{S}} \cup \{g_I(v'') : 1\} \models VC[V/v'']$. We formally define this notion below.

Definition 5.14 (A-Priori VC). A SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$ has an **A-Priori VC** w.r.t. a given social network $\mathscr{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ and a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$ iff for each $\mathbf{V}' \subseteq \mathbf{V}$ the following holds: for each $v'' \in \mathbf{V}'$, $\Pi \cup \{g_I(v') : 1 \mid v' \in \mathbf{V}'\} \models VC[V/v'']$ iff $\Pi_{\mathscr{S}} \cup \{g_I(v'') : 1\} \models VC[V/v'']$.

If, in the cell phone example, we require that the free cell phones are given to "good" vertices, then query **(Q1)** is a-priori VC because being "good" may be defined statically and is not determined by the diffusion process. Likewise, if we consider our medical example, in the case of an a-priori VC query **(Q2)** saying that an individual below 5 should not get the medicine, this boils down to a static labeling of each node's age (below 5 or not) which is not affected by the diffusion process. Table 5.2 summarizes the different properties that we prove in this section (as well as what assumptions we make to prove these properties).

We say that function *value* is monotonic iff $\mathbf{V}_1 \subseteq \mathbf{V}_2$ implies $value(\mathbf{V}_1) \leq value(\mathbf{V}_2)$ for any two sets of vertices $\mathbf{V}_1$ and $\mathbf{V}_2$. The first property we show is that the value function is monotonic if *agg* is monotonic.

Lemma 5.1. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathscr{S}$, and a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$, if agg is monotonic (Definition 5.11), then value (defined as per Q and Π) is monotonic.*

Before introducing the next result we recall the definitions of matroid and uniform matroid. A *matroid* is a pair (X, I) where X is a finite set and I is a collection of subsets of X (called "independent"), satisfying two axioms:

1. $B \in I, A \subset B \Rightarrow A \in I$.
2. $A, B \in I, |A| < |B| \Rightarrow \exists x \in B - A \ s.t. \ A \cup \{x\} \in I$.

A *uniform* matroid is a matroid such that independent sets are all sets of size at most k for some $k \geq 1$.

Next, we show that the set of pre-answers is a uniform matroid in the special case of an a-priori VC query.

Lemma 5.2. *Given a SNDOP query* $Q = (agg, VC, k, g_I(V), g_O(V))$*, a social network* $\mathscr{S}$*, and a GAP* $\Pi \supseteq \Pi_{\mathscr{S}}$*, if* Q *is a-priori VC w.r.t.* $\mathscr{S}$ *and* Π*, then the set of pre-answers is a uniform matroid.*

As we will see in Sect. 5.5, the above lemma (along with other properties, see Theorem 5.5) enables us to define a greedy approximation algorithm to solve SNDOP queries that achieves the best possible approximation ratio that a polynomial algorithm can achieve (unless $\mathbf{P} = \mathbf{NP}$).

An important property in social networks is *submodularity* whose relationship to the spread of phenomena in social networks has been extensively studied (see Chap. 4). If X is a set, then a function $f : 2^X \to \mathbb{R}$ is *submodular* iff whenever $X_1 \subseteq X_2 \subseteq X$ and $x \in X - X_2$, $f(X_1 \cup \{x\}) - f(X_1) \geq f(X_2 \cup \{x\}) - f(X_2)$. The following result states that the *value* function associated with a linear GAP and an a-priori VC SNDOP query whose aggregate is positive-linear is guaranteed to be submodular.

Theorem 5.1. *Given an SNDOP query* $Q = (agg, VC, k, g_I(V), g_O(V))$*, a social network* $\mathscr{S}$*, and a GAP* $\Pi \supseteq \Pi_{\mathscr{S}}$*, if the following criteria are met:*

- Π *is a linear GAP,*
- Q *is a-priori VC, and*
- *agg is positive-linear,*

then value (defined as per Q *and* Π*) is* ***sub-modular****.*

5.3.3 The Complexity of SNDOP Queries

We now study the complexity of answering an SNDOP query. First, we show that SNDOP query answering is NP-hard by a reduction from max k-cover [14]. We show that the problem is NP-hard even when many of the special cases hold.

Theorem 5.2. *Finding an answer to an SNDOP query* $Q = (agg, VC, k, g_I(V), g_O(V))$ *(w.r.t. a social network* $\mathscr{S}$ *and a GAP* $\Pi \supseteq \Pi_{\mathscr{S}}$*) is NP-hard (even if* Π *is a linear GAP,* $VC = \emptyset$*, agg = SUM and value is normalized).*

Under some conditions, the decision problem for SNDOP queries is also in NP.

Theorem 5.3. *Given a SNDOP query* $Q = (agg, VC, k, g_I(V), g_O(V))$*, a social network* $\mathscr{S}$*, a GAP* $\Pi \supseteq \Pi_{\mathscr{S}}$*, and a real number target, the problem of checking whether there exists a pre-answer* $\mathbf{V'}$ *s.t. value*$(\mathbf{V'}) \geq$ *target is in NP under the assumptions that agg and the functions in* $\mathscr{F}$ *are polynomially computable, and* Π *is ground.*

Most common aggregate functions like SUM, AVERAGE, Weighted average, MIN, MAX, COUNT are all polynomially computable. Moreover, the assumption that the functions corresponding to the function symbols in $\mathscr{F}$ (i.e. the function symbols that can appear in the annotations of a GAP) are polynomially computable is also reasonable.

Later in this chapter, we shall address the problem of answering a SNDOP query using an approximation algorithm. We say that $\mathbf{V}'$ is a $\frac{1}{\alpha}$-approximation to an SNDOP query if $value(\mathbf{V}_{opt}) \leq \alpha \cdot value(\mathbf{V}')$ (where $\mathbf{V}_{opt}$ is an answer to the SNDOP query). Likewise, the algorithm that produces $\mathbf{V}'$ in this case is an α-approximation algorithm. We note that due to the nature of the reduction from MAX-K-COVER that we used to prove NP-hardness, we can now show that unless $\mathbf{P} = \mathbf{NP}$, there is no PTIME-approximation of the SNDOP query answering problem that can guarantee that the approximate answer is better than 0.63 of the optimal value. This gives us an idea of the limits of approximation possible for a SNDOP query with a polynomial-time algorithm. Later, we will develop a greedy algorithm that precisely matches this approximation ratio.

Theorem 5.4. *Answering a SNDOP query* $Q = (agg, VC, k, g_I(V), g_O(V))$ *(w.r.t. a social network* $\mathscr{S}$ *and a GAP* $\Pi \supseteq \Pi_{\mathscr{S}}$*) cannot be approximated in PTIME within a ratio of* $\frac{e-1}{e} + \varepsilon$ *for some* $\varepsilon > 0$ *(where e is the inverse of the natural log) unless* $\mathbf{P} = \mathbf{NP}$*—even if* Π *is a linear GAP,* $VC = \emptyset$*,* $agg = SUM$ *and value is normalized.*

In other words, the previous theorem says that there is no polynomial-time algorithm that can approximate *value* within a factor of about 0.63 under standard assumptions.

5.4 Applying SNDOPs to Diffusion Problems

In this section, we show how SNDOPs can be applied to real-word diffusion problems. Most diffusion models in the literature fall into one of three categories—**tipping** models (Sect. 5.4.1), where a given vertex adopts a behavior based on the ratio of how many of its neighbors previously adopted the behavior, **cascade** models (Sect. 5.4.2), where a property passes from vertex to vertex solely based on the strength of the relationship between the vertices, and **homophilic** models (Sect. 5.4.3), where vertices with similar properties tend to adopt the same behavior—irrespective (or in addition to) of network relationships.

5.4.1 Tipping Diffusion

Tipping models [6, 20, 21] have been studied extensively in economics and sociology to understand diffusion phenomena. In tipping models, a vertex changes a property based on the cumulative effect of its neighbors. In this section, we present the tipping model of Jackon and Yariv [10], which generalizes many existing tipping models.

The Jackson-Yariv Diffusion Model [10] In this framework, the social network is just an undirected graph $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$ consisting of a set of agents (e.g. people). Each agent has a default behavior (A) and a new behavior (B). Suppose d_i denotes the degree of a vertex v_i. Jackson and Yariv [10] use a function $\gamma : \{0, \ldots, |\mathbf{V}'| - 1\} \to [0,1]$ to describe how the number of neighbors of v affects the benefits to v for adopting behavior B. For instance, $\gamma(3)$ specifies the benefits (in adopting behavior B) that accrue to an arbitrary vertex $v \in \mathbf{V}'$ that has three neighbors. Let π_i denote the fraction of neighbors of v_i that have adopted behavior B. Let constants b_i and ρ_i be the agent-specific benefit and cost, respectively, for vertex v_i to adopt behavior B. Jackson and Yariv [10] state that node v_i switches to behavior B iff $\frac{b_i}{\rho_i} \cdot \gamma(d_i) \cdot \pi_i \geq 1$.

Returning to our cell-phone example, one could potentially use this model to describe the spread of the new plan. In this case, behavior A would be adherence to the current plan the user subscribes to, while B would be the use of the new plan. The associated SNDOP query would find a set of nodes which, if given a free plan, would jointly maximize the expected number of adoptees of the plan. Cost and benefit could be computed from factors such as income, time invested in switching plans, etc. We show how the model of [10] can be expressed via GAPs.

Diffusion Model 5.4.1 (Jackson-Yariv model) *Given a Jackson-Yariv model consisting of $\mathbf{G}' = (\mathbf{V}', \mathbf{E}')$, we can set up a social network $\mathscr{S} = (\mathbf{V}', \mathbf{E}'', \ell_{vert}, \ell_{edge}, w)$ as follows. We define $\mathbf{E}'' = \{(x,y),(y,x) \mid (x,y) \in \mathbf{E}'\}$. We have a single edge predicate symbol edge which is assigned by ℓ_{edge} to every edge in $\mathbf{E}''$, and w assigns 1 to all edges in $\mathbf{E}''$. Our associated GAP Π_{JY} now consists of $\Pi_{\mathscr{S}}$ plus one rule of the following form for each vertex v_i:*

$$B(v_i) : \left\lfloor \frac{b_i}{\rho_i} \cdot \gamma\left(\sum_j E_j\right) \cdot \frac{\sum_j X_j}{\sum_j E_j} \right\rfloor \leftarrow \bigwedge_{v_j \mid \langle v_j, v_i \rangle \in \mathbf{E}''} (edge(v_j, v_i) : E_j \wedge B(v_j) : X_j)$$

It is easy to see that this rule (when applied in conjunction with $\Pi_{\mathscr{S}}$ for a social network $\mathscr{S}$) precisely encodes the Jackson-Yariv semantics.

5.4.2 Cascading Diffusion

In a **cascading** model, a vertex obtains a property from one of its neighbors, typically based on the strength of its relationship with the neighbor. These models were introduced in the epidemiology literature and perhaps the most well-known of these models, the SIR model, is more fully reviewed in Chap. 2. These cascading diffusion models have been applied to other domains as well. For example, Cha et al. [9] (diffusion of photos in Flickr) and Sun et al. [8] (diffusion of bookmarks in Facebook) both look at diffusion process in social networks as "social cascades" of this type.

The SIR Model of Disease Spread The SIR (*susceptible, infectious, removed*) model of disease spread (see Chap. 2) is a classic disease model which labels each vertex in a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ (of humans) with *susceptible* if it has not had the disease but can receive it from one of its neighbors, *infectious* if it has caught the disease and t_{rec} units of time have not expired, and *removed* when the vertex can no longer catch or transmit the disease. The SIR model assumes that a vertex v that is infected can transmit the disease to any of its neighbors v' with a probability $p_{v,v'}$ for t_{rec} units of time. It is assumed that becoming infected takes precisely a time unit. We would like to find a set of at most k vertices that would maximize the expected number of vertices that become infected. These are obviously good candidates to treat with appropriate medications. The following is a non-probabilistic variant of the SIR model represented as a GAP. Note it is not equivalent to the SIR model of Chap. 2—though it captures the intuition.

Diffusion Model 5.4.2 (SIR Model) *Let $\mathscr{S} = (\boldsymbol{V}, \boldsymbol{E}, \ell_{vert}, \ell_{edge}, w)$ be an SN where each edge is labeled with the predicate symbol e and $w(\langle v, v' \rangle) = p_{v,v'}$ assigns a probability of transmission to each edge . We use the predicate inf to designate the initially infected vertices. In order to create a GAP Π_{SIR} capturing the SIR model of disease spread, we first define t_{rec} predicate symbols $rec_1, \ldots, rec_{t_{rec}}$ where $rec_i(v)$ intuitively means that node v was infected i days ago. Hence, $rec_{t_{rec}}(v)$ means that v is "removed." We embed $\mathscr{S}$ into GAP Π_{SIR} by adding the following diffusion rules. If $t_{rec} > 1$, we add a non-ground rule for each $i = \{2, \ldots, t_{rec}\}$ - starting with t_{rec}:*

$$rec_i(V) : R \leftarrow rec_{i-1}(V) : R$$

$$rec_1(V) : R \leftarrow inf(V) : R$$

$$inf(V) : (1-R) \cdot P_{V',V} \cdot P_{V'} \cdot (1-R') \leftarrow rec_{t_{rec}}(V) : R \wedge e(V',V) : P_{V',V} \wedge inf(V') : P_{V'} \wedge rec_{t_{rec}}(V') : R'.$$

The first rule says that if a vertex is in its $(i-1)$'th day of recovery with confidence R in the j'th iteration of the $\mathbf{T}_{\Pi_{SIR}}$ operator, then the vertex is i days into recovery (with the same confidence) in the $j+1$'th iteration of the operator. Likewise, the second rule intuitively encodes the fact that if a vertex became infected with confidence R in the j'th iteration of the $\mathbf{T}_{\Pi_{SIR}}$ operator, then the vertex is one day into recovery in the $j+1$'th iteration of the operator with the same confidence. The last rule says that if a vertex V' was infected with confidence $P_{V'}$ and has not been removed with confidence $1-R'$, and there is an edge from V' to V in the social network (weighted with $P_{V',V}$), given the confidence $1-R$ that V has not already been removed, then the confidence that the vertex V gets infected is $(1-R) \cdot P_{V',V} \cdot P_{V'}(1-R')$. Here, $P_{V'}(1-R')$ is one way of measuring the confidence that V' is infected and has not recovered and $P_{V',V}$ is the confidence of infecting the neighbor. Notice that e is a static property of the graph which does not change over the time, so we do not need time indexes for it. As for *inf*, the reason why we can avoid using time indexes is that we can keep track of how much time has gone since a vertex got infected with the predicates rec_i using the second rule.

Diffusion in the Flickr Photo Sharing Network The Flickr social network allows users to share photographs. Users create a list of “favorite” photos that can be viewed by other users. Cha et al. [9] use a variant of SIS above to study how photographs spread to the favorite lists of different users. A key difference is that they do not consider a node “recovered”—i.e. once a photo was placed on a favorite list, it was relatively permanent (the study was conducted over about 100 days). They also found that photos lower on a favorite list (as the result of a user marking a large number of photos as “favorite”) for a given user could still spread through the network. A simple GAP that captures the intuition of how Flickr photos spread according to [9] uses just one rule:

Diffusion Model 5.4.3 (Flickr Photo Diffusion)

$$photo_i(V) : const_i \cdot X_i \leftarrow connected\_to(V',V) : 1 \wedge photo_i(V') : X_i$$

In Diffusion Model 5.4.3, the annotation of the vertex atom $photo_i(V)$ is the confidence that vertex V has marked photo i as one of its favorites. The predicate *connected_to* is the sole edge label representing that there is a connection from vertex V' to V (users select other users on this network). Additionally, the value $const_i$ is a number in $[0,1]$ that determines how a given photo spreads in the network. Notice that the above rule is linear, as the head is a linear combination and $const_i \in [0,1]$. We note that for all of these models, the annotation functions reflect one interpretation of the confidence that a vertex is infected or recovered—others are possible in our framework.

5.4.3 *Homophilic Diffusion*

Recently, [11] studied the spread of mobile application use on a global instant-messaging network of over 27 million vertices. They found that network-based diffusion could overestimate the spread of a mobile application and, for this scenario, over 50 % of the adopted use of the applications was due to **homophily**—vertices with similar properties adopting similar applications.

These results should not be surprising—the basic idea behind web-search advertising is that two users with a similar property (search term) will be interested in the same advertised item. In fact, Cha et al. [9] explicitly pre-processed their Flickr data set with a heuristic to *eliminate* properties attached to vertices that could not be accounted for by a diffusion process. We can easily represent homophilic diffusion in a GAP with the following type of diffusion rule:

Diffusion Model 5.4.4 (Homophilic Diffusion of a Product)

$$buys\_product(V) : 0.5 \times X \leftarrow property(V) : 1 \wedge exposed\_to\_product(V) : X$$

In Diffusion Model 5.4.4, if a vertex is exposed to a product (e.g. through mass advertising) and has a certain property, then the person associated with the vertex purchases the product with a confidence of $0.5 \times X$, where X measures the extent of the exposure. For this rule, there are no network effects.

In [10], the authors propose a "big seed" marketing approach that combines both homophilic and network effects. They outline a strategy of advertising to a large group of individuals who are likely to spread the advertisement further through network effects. We now describe a GAP that captures the ideas underlying big seed marketing. Suppose we have a set of vertex predicate symbols $\mathsf{AL} \subseteq \mathsf{VP}$ corresponding to people "attributes"—these may be certain demographic characteristics such as education level, race, level of physical fitness, etc. Suppose we want to advertise to people having (at least) one of $k \leq |\mathsf{AL}|$ attributes to maximize an aggregate *agg* with respect to a goal predicate g (in other words, we want to choose k attributes and advertise to people having those attributes so that *agg* with respect to g is maximized). Consider the following construction.

Diffusion Model 5.4.5 (Big Seed Marketing) *The GAP includes an embedding of the social network as well as the network diffusion model of the user's choice. We make the following additions to the GAP and the SN:*

1. *Add vertex predicate symbol attrib to* VP.
2. *For each* $lbl \in \mathsf{AL}$, *add a vertex* v_{lbl} *to* $\mathbf{V}$. *We also set* $\ell_{vert}(v_{lbl}) = \{attrib\}$.
3. *For each* $lbl \in \mathsf{AL}$, *add the following non-ground rule:*

$$g(V) : \mathit{eff}_{lbl} \times X \leftarrow lbl(V) : 1 \wedge g(v_{lbl}) : X$$

where eff_{lbl} *is a constant in* $[0,1]$ *corresponding to the confidence that, if advertised to, a vertex* v *with attribute* lbl *obtains an annotation of* 1 *on* $g(v)$.

Our SNDOP query is $(agg, VC, k, g(V), g(V))$, *where* $VC = \{attrib(V) : 1\}$.

Note that in the above diffusion model, the v_{lbl} vertices correspond to advertisements directed toward different vertex properties. The VC condition forces the query to only return v_{lbl} vertices. As an example, a solution like $\{g(v_{lbl_1}), g(v_{lbl_2})\}$ means that we are targeting people having attribute lbl_1 or lbl_2. The diffusion rule, added per label, ensures that the mass advertisement is received and that the vertex acts accordingly (hence the eff_{lbl} constants).

5.5 Algorithmic Approach and Experiments

Regretfully, Theorem 5.2 precludes an exact solution in PTIME and Theorem 5.4 precludes a PTIME α-approximation algorithm where $\alpha < \frac{e}{e-1}$. Both of these results hold for the restricted case of linear-GAPs and positive-linear aggregate functions.

The good news is that we were able to show that (1) for linear-GAPs and a-priori VC queries with positive-linear aggregates, the *value* function is *submodular* (Theorem 5.1). (2) Under these conditions, we can reduce the problem to the maximization of a submodular function over a uniform matroid (the uniformity of the matroid is proved in Lemma 5.2 for a-priori VC queries). (3) We can leverage the work of [13] that admits a greedy $\frac{e}{e-1}$ approximation algorithm. In this section, we develop a greedy algorithm for SNDOP queries that leverages the above three results. This is analogous to the greedy approximation technique for the IC and LT models described in Chap. 4.

The GREEDY-SNDOP algorithm shown below assumes a linear GAP, an a-priori VC query with positive-linear aggregates, and a Normalized *value* function (notice that the latter requirement can be met as stated by Proposition 5.2). The algorithm provides $\frac{e}{e-1}$ approximation to the SNDOP query problem. As this matches the upper bound of Theorem 5.4, we cannot do better in terms of an approximation guarantee.

```
GREEDY-SNDOP$(\Pi, agg, VC, k, g_I(V), g_O(V))$ returns $SOL \subseteq \mathbf{V}$

1. Initialize $SOL = \emptyset$ and $REM = \{v \in \mathbf{V} | \left(\{g_I(v):1\} \cup \bigcup_{pred \in \ell_{vert}(v)} \{pred(v):1\}\right) \models VC[V/v]\}$
2. While $|SOL| < k$ and $REM \neq \emptyset$
    a. $v_{best} = \text{null}$, $val = value(SOL)$, $inc = 0$
    b. For each $v \in REM$, do the following
        i. Let $inc_{new} = value(SOL \cup \{v\}) - val$
        ii. If $inc_{new} \geq inc$ then $inc = inc_{new}$ and $v_{best} = v$
    c. $SOL = SOL \cup \{v_{best}\}$, $REM = REM - \{v_{best}\}$
3. Return $SOL$
```

We now analyze the time complexity of GREEDY-SNDOP.

Proposition 5.3. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathscr{S}$, and a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$, the complexity of GREEDY-SNDOP is $O(k \cdot |\mathbf{V}| \cdot F(|\mathbf{V}|))$ where $F(|\mathbf{V}|)$ is the time complexity to compute value$(\mathbf{V}')$ for some set $\mathbf{V}' \subseteq \mathbf{V}$ of size k.*

We note that most likely, the most expensive operation is the computation of *value* at line 2(b)i. One obvious way to address this issue is by using a non-ground version of the fixed-point.

Theorem 5.5. *Given a SNDOP query $Q = (agg, VC, k, g_I(V), g_O(V))$, a social network $\mathscr{S}$, and a GAP $\Pi \supseteq \Pi_{\mathscr{S}}$, if*

- *Π is a linear GAP,*
- *Q is a-priori VC,*
- *agg is positive-linear, and*
- *value is Normalized,*

then GREEDY-SNDOP is an $(\frac{e}{e-1})$-approximation algorithm.

We have implemented the GREEDY-SNDOP algorithm in 660 lines of Java code by re-using and extending the diffusion modeling Java library of [2] (approx 35 K lines of code). Our implementation uses multiple threads in the inner loop of the GREEDY-SNDOP algorithm to increase efficiency. All experiments were executed on the same machine with a dedicated 4-core 2.4 GHz processor and 22 GB of main memory. Times were measured to millisecond precision and are reported in seconds.

Data Set In order to evaluate GREEDY-SNDOP, we used a real-world dataset based on a social network of Wikipedia administrators and authors. Wikipedia is an online encyclopedia collaboratively edited by many contributors from all over the world. Selected contributors are given privileged administrative access rights to help maintain and control the collection of articles with additional technical features. A vote by existing administrators and ordinary authors determines whether an individual is granted administrative privileges. These votes are publicly recorded. Leskovec et al. [3] crawled 2794 elections from the inception of Wikipedia until January 2008. The votes casted in these elections give rise to a social network among Wikipedia administrators and authors by representing a vote of user i for user j as a directed edge from node i to j. In total, the dataset contains 103,663 votes (edges) connecting more than 7000 Wikipedia users (vertices). Hence, the network is large and densely connected.[9]

SNDOP-Query In our experiments, we consider the hypothetical problem of finding a set of administrators having the highest overall influence in the Wikipedia social network described above. We treat votes as a proxy for the inverse of influence. In other words, if user i voted for user j, we assume user j (intentionally through lobbying or unintentionally through the force of his contributions to Wikipedia) influenced user i to vote for him. All edges are assigned a weight of 1. Our SNDOP queries are designed as per the following definition.

Definition 5.15 (Wikipedia SNDOP-Query). Given some natural number $k > 1$, a Wikipedia SNDOP query, $WQ(k)$ is specified as follows:

- $agg = \mathsf{SUM}$—the intuition is that the aggregate provides us an expected number of vertices that are influenced.
- $VC = \emptyset$—we do not use a vertex condition in our experiments
- k as specified by the input
- $g_I(V) = g_O(V) = \mathit{influenced}(V)$

Diffusion Models Used We represented the diffusion process with two different models: one tipping and one cascading.

- **Cascading Diffusion Model** We used the Flickr Diffusion Model (Diffusion Model 5.4.3 on page 66) described in Sect. 5.4.2. In this model, a constant parameter α represents the "strength" or "likelihood" of influence. The larger the parameter α the higher the influence of a user on those who voted for her.

[9]Our Wikipedia data set does not include edge weights. However, including edge weights should not appreciably change the experimental results which show that solving SNDOP queries when tipping models are used is faster, in general, than when cascade models are used.

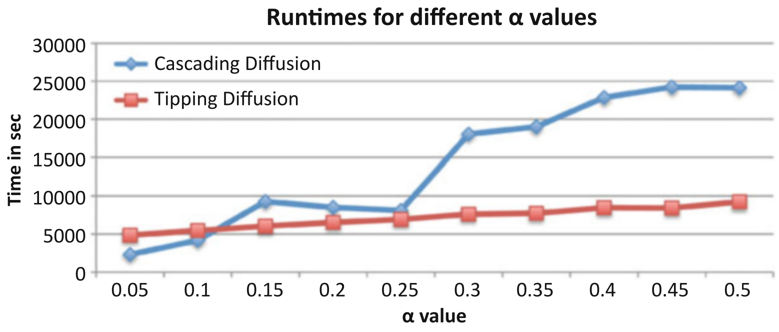

Fig. 5.2 Runtimes of GREEDY-SNDOP for different values of α and $k = 5$ in both diffusion models

- **Tipping Diffusion Model** Cha et al. [5] shows that there is a relationship between the likelihood of a vertex marking a photo as a favorite and the percentage of their neighbors that also marked that photo as a favorite. This implies a tipping-model (as in Sect. 5.4.1). We apply the Jackson-Yariv model with B equated to *influenced*. For each vertex $v_j \in \mathbf{V}$, we set the benefit to cost ratio $(\frac{b_j}{c_j})$ to 1. Finally, the function γ defined in the Jackson Yariv model is the constant-valued function (for all values of x):

$$\gamma(x) = \alpha.$$

  This says that irrespective of the number of neighbors that a vertex has, the benefit to adopting strategy B (i.e. *influenced*) is α. Therefore, the resulting diffusion rule for the linear Jackson-Yariv model is:

$$\mathit{influenced}(v) : \alpha \cdot \frac{\sum_j X_j}{|\{v_j | \langle v_j, v\rangle\}|} \leftarrow \bigwedge_{v_j | \langle v_j, v\rangle \in \mathbf{E}} \mathit{influenced}(v_j) : X_j$$

For both models, we derive a unique logic program for each setting of the parameter α. The parameter α depends on the application and can be learned from ground truth data. In our experiments, we varied α to avoid introducing bias.

Run-Time of GREEDY-SNDOP with Varying α and Different Diffusion Models Figure 5.2 shows the total runtime of GREEDY-SNDOP in seconds to find the set of $k = 5$ most influential users in the Wikipedia voting network for different values of the strength of influence parameter α. We varied α from 0.05 (very low level of influence) to 0.5 (very high level of influence) for both the cascading and tipping diffusion model. We observe that higher values of α lead to higher runtimes as expected since the scope of influence of any individual in the network is larger. Furthermore, we observe that the runtimes for the tipping diffusion model increase more slowly with α compared to the cascading model.

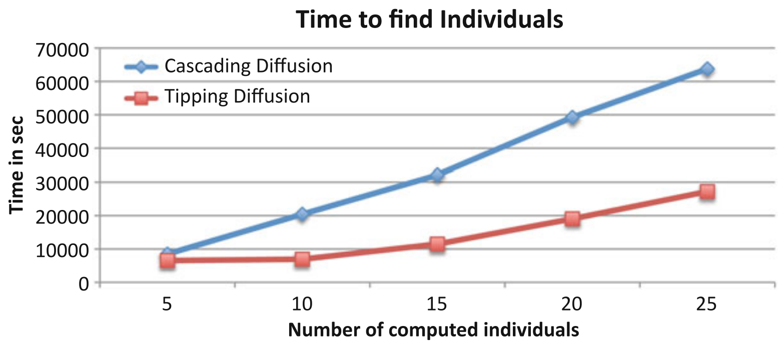

Fig. 5.3 Runtimes of GREEDY-SNDOP for different values of k and $\alpha = 0.2$ in both diffusion models

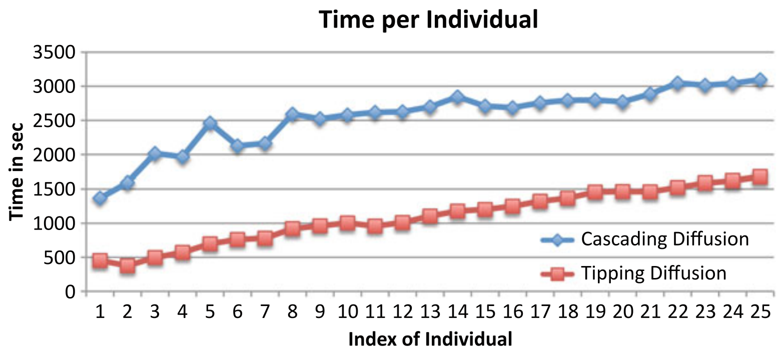

Fig. 5.4 Time per iteration of GREEDY-SNDOP for $\alpha = 0.2$ in both diffusion models

Run-Time of GREEDY-SNDOP with Varying k For the next set of experiments, we keep the strength of influence fixed to $\alpha = 0.2$ and varied k which governs the size of the set of influencers. Figure 5.3 reports the runtime of GREEDY-SNDOP for the query $WQ(k)$ with $k = 5, 10, 15, 20, 25$. For the cascading model, the runtime is approximately linear in k—a curve-fitting analysis using Excel showed a slight superlinear trend (even though the figure itself looks linear at first sight). Figure 5.4 shows the time taken to execute each of the 25 iterations of the outer loop for the query $WQ(25)$ with $\alpha = 0.2$. Note that each subsequent iteration is more expensive than the previous one since the size of the logic programs to consider increases with the addition of each ground atom $\mathit{influenced}(v_i)$. However, we also implemented the practical improvement of "lazy evaluation" of the submodular function as described in [7]. This improvement, which maintains correctness of the algorithms, stores previous improvements in total score and prunes the greedy search for the highest scoring vertex as discussed. We found that this technique also reduced the runtime of subsequent iterations.

Our experimental results show that we can answer SNDOP queries on large social networks. For example, computing the set of five most influential Wikipedia users in the voting network required approximately 2 h averaged over the different values of α in the tipping diffusion model.

5.6 Conclusion

Social networks are proliferating rapidly and have led to a wave of research on diffusion of phenomena in social networks. In this chapter, we introduce the class of *Social Network Diffusion Optimization Problems* (SNDOPs for short) which try to find a set of vertices (where each vertex satisfies some user specified vertex condition) that has cardinality k or less (for a user-specified $k > 0$) and that optimizes an objective function specified by the user in accordance with a diffusion model represented via the well-known generalized annotated program (GAP) framework. We have used specific examples of SNDOP queries drawn from product adoption (cell phone example) and epidemiology.

References

1. Subrahmanian, Venkatramana S., and Diego Reforgiato. "AVA: Adjective-verb-adverb combinations for sentiment analysis." Intelligent Systems, IEEE 23.4 (2008): 43–50.
2. Broecheler, Matthias, Paulo Shakarian, and V. S. Subrahmanian. "A scalable framework for modeling competitive diffusion in social networks." Social Computing (SocialCom), 2010 IEEE Second International Conference on. IEEE, 2010.
3. Leskovec, Jure, Daniel Huttenlocher, and Jon Kleinberg. "Predicting positive and negative links in online social networks." Proceedings of the 19th international conference on World wide web. ACM, 2010.
4. Shakarian, Paulo, et al. "Using generalized annotated programs to solve social network diffusion optimization problems." ACM Transactions on Computational Logic (TOCL) 14.2 (2013): 10.
5. Cha, Meeyoung, Alan Mislove, and Krishna P. Gummadi. "A measurement-driven analysis of information propagation in the flickr social network." Proceedings of the 18th international conference on World wide web. ACM, 2009.
6. Kifer, M., Subrahmanian, V.S., "Theory of generalized annotated logic programming and its applications." The Journal of Logic Programming, 12(4), 1992.
7. Leskovec, Jure, et al. "Cost-effective outbreak detection in networks." Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.
8. Sun, Eric, et al. "Gesundheit! Modeling Contagion through Facebook News Feed." ICWSM. 2009.
9. Cha, Meeyoung, et al. "Characterizing social cascades in flickr." Proceedings of the first workshop on Online social networks. ACM, 2008.
10. Watts, Duncan J., Jonah Peretti, and Michael Frumin. Viral marketing for the real world. Harvard Business School Pub., 2007.

11. Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks
12. Goundan, Pranava R., and Andreas S. Schulz. "Revisiting the greedy approach to submodular set function maximization." Optimization online (2007): 1–25.
13. Nemhauser, George L., Laurence A. Wolsey, and Marshall L. Fisher. "An analysis of approximations for maximizing submodular set functions." Mathematical Programming 14.1 (1978): 265–294.
14. Feige, Uriel. "A threshold of ln n for approximating set cover." Journal of the ACM (JACM) 45.4 (1998): 634–652.
15. Hethcote, Herbert W. "Qualitative analyses of communicable disease models." Mathematical Biosciences 28.3 (1976): 335–356.
16. Cowan, Robin, and Nicolas Jonard. "Network structure and the diffusion of knowledge." Journal of economic Dynamics and Control 28.8 (2004): 1557–1575.
17. Watts, Duncan J. "Networks, dynamics, and the small-world phenomenon 1." American Journal of sociology 105.2 (1999): 493–527.
18. Rychtár, Jan, and Brian Stadler. "Evolutionary dynamics on small-world networks." International Journal of Computational and Mathematical Sciences 2.1 (2008): 1–4.
19. Lloyd, J. Foundations of Logic Programming. Berlin: Springer-Verlag, 1987.
20. Granovetter, Mark. "Threshold models of collective behavior." American journal of sociology (1978): 1420–1443.
21. Schelling, Thomas C. Micromotives and macrobehavior. WW Norton & Company, 2006.
22. Anderson, Roy M., and Robert M. May. "Population biology of infectious diseases: Part I." Nature 280 (1979): 361–7.
23. P. Shakarian, V.S. Subrahmanian, M.L. Sapino. Using Generalized Annotated Programs to Solve Social Network Optimization Problems. 26th Intl. Conference on Logic Programming (ICLP-10) (Jul. 2010).
24. Christoff, Z., Hansen, J.U., A logic for diffusion in social networks. Journal of Applied Logic, 13(1), 2015.
25. P. Shakarian, G.I. Simari, D. Callahan. Reasoning about Complex Networks: A Logic Programming Approach. 29th Intl. Conference on Logic Programming (ICLP-13) (Aug. 2013).
26. Kang, C., Molinaro, C., Kraus, S., Shavitt, Y., Subrahmanian, V.S. Diffusion Centrality in Social Networks. IEEE ASONAM, 2012.
27. Coelho, Flávio C., Oswaldo G. Cruz, and Cláudia T. Codeço. "Source Code for Biology and Medicine." Source code for biology and medicine 3 (2008): 3.