Technical Communications of the International Conference on Logic Programming, 2010 (Edinburgh), pp. 182–191
http://www.floc-conference.org/ICLP-home.html

USING GENERALIZED ANNOTATED PROGRAMS TO SOLVE SOCIAL NETWORK OPTIMIZATION PROBLEMS

PAULO SHAKARIAN [1] AND V.S. SUBRAHMANIAN [1] AND MARIA LUISA SAPINO [2]

[1] Univestiy of Maryland
College Park, MD
E-mail address: {pshak,vs}@cs.umd.edu

[2] Università di Torino
Torino, Italy
E-mail address: mlsapino@di.unito.it

Abstract. Reasoning about social networks (labeled, directed, weighted graphs) is becoming increasingly important and there are now models of how certain phenomena (e.g. adoption of products/services by consumers, spread of a given disease) "diffuse" through the network. Some of these diffusion models can be expressed via generalized annotated programs (GAPs). In this paper, we consider the following problem: suppose we have a given goal to achieve (e.g. maximize the expected number of adoptees of a product or minimize the spread of a disease) and suppose we have limited resources to use in trying to achieve the goal (e.g. give out a few free plans, provide medication to key people in the SN) - how should these resources be used so that we optimize a given objective function related to the goal? We define a class of *social network optimization problems* (SNOPs) that supports this type of reasoning. We formalize and study the complexity of SNOPs and show how they can be used in conjunction with existing economic and disease diffusion models.

1. Introduction

There is a rapid proliferation of different types of *graph data* in the world today. These include social network data (FaceBook, Flickr, YouTube, etc.), cell phone network data [NE08] collected by virtually all cell phone vendors, email network data (such as those derived from the Enron corpus or Gmail logs), as well as information on disease networks [FC08, And79]. In addition, the World Wide Consortium's RDF standard is also a graph-based standard for encoding semantic information contained in web pages. There has been years of work on analyzing how various properties of nodes in such networks "diffuse" through the network - different techniques have been invented in different academic disciplines including economics [Jac05, Sch78], infectious diseases [FC08], sociology [Gra78] and computer science [Kem03].

1998 ACM Subject Classification: I.2.4 Knowledge Representation Formalisms and Methods.
Key words and phrases: annotated logic programming, optimization queries, social networks.
Some of the authors of this paper were funded in part by AFOSR grant FA95500610405, ARO grant W911NF0910206 and ONR grant N000140910685.

Technical Communications of the 26th International Conference on Logic Programming, Edinburgh, July, 2010
Editors: Manuel Hermenegildo, Torsten Schaub
LIPIcs - Leibniz International Proceedings in Informatics. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/LIPIcs.ICLP.2010.182

Many of these methods focus on modeling a *specific* type of diffusion in an SN and often, they only rely on the network topology [Wat99, Cow04, Ryc08], rather than on properties of vertices, and the nature of the relationships between vertices. In this paper, we first argue that Generalized Annotated Programs (GAPs) [Kif92b, Kif92a, Thi93] and their variants [Ven04, Kra04, Lu96, Lu93, Dam99] form a convenient method to express many diffusion models. Next, unlike most existing work in social networks which focus on learning diffusion models, we focus on reasoning with previously learned diffusion models (expressed via GAPs). In particular, if we wish to achieve certain *goals* based on a social network, how best can we achieve these goals? Two examples are given below.

- **(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. Which k people should they pick so as to maximize the (expected) 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 the k people who maximally spread the disease (so that they can provide immediate treatment to these k people in an attempt to halt the disease's spread).

Both the above problems are instances of a class of queries that we call SNOP queries. They differ from queries studied in the past in quantitative (both probabilistic and annotated) logic programming in two fundamental ways: (i) They are specialized to operate on graph data, (ii) They optimize complex kinds of objective functions. Neither of these has been studied before by any kind of quantitative logic programming framework, though work on optimizing objective functions in the context of different types of semantics (minimal model and stable model semantics) has been studied before[Leo04]. And of course, constraint logic programming[Apt03] has also extensively studied optimization issues as well in logic programming - however, here, optimization and constraint solving is embedded in the constraint logic program, whereas in our case, they are part of the query over an annotated logic program.

This paper is organized as follows. In Section 2, we provide an overview of GAPs (past work), define a social network, and explain how GAPs can represent some types of diffusion in SNs. Section 3 formally defines different types of social network optimization problems and provides results on their computational complexity. Finally, section 4 shows how our framework can represent several existing diffusion models for social networks including one each from economics, epidemiology, and computer science.

2. Technical Preliminaries

In this section, we first formalize social networks, then briefly overview generalized annotated logic programs (GAPs)[Kif92b] and then describe how GAPs can be used to represent concepts related to diffusion in SNs. Throughout this paper, 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 2.1. A **social network** ($\mathcal{S}$) is a 5-tuple $(\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ where:

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

We now present a brief example of an SN that will be used throughout this paper.

Example 2.2. Let us return to the cell phone example (query **(Q1)**). Figure 1 shows a toy SN the cell phone company might use. Here, we might have $\mathsf{VP} = \{male, female, adopters, temp\_adopter, non\_adptr\}$ 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. $w(v_1, v_2, ep)$ denotes the percentage of communications of type $ep \in \mathsf{EP}$ initiated by v_1 that were with v_2 (measured either w.r.t. time or bytes). The function ℓ_{vert} is shown in the figure 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 illustrate ℓ_{edge}.

It is important to note that our definition of social networks is much broader than that used by several researchers[And79, FC08, Jac05, Kem03] who often do not consider either ℓ_{edge} or ℓ_{vert} — these can have a significant impact on what we do with such networks.

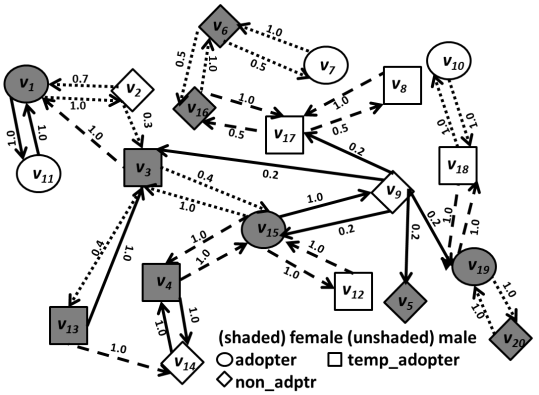


Figure 1: Example cellular network.

Note. We note that each social network must satisfy various integrity constraints. In Example 2.2, it is clear that $\ell_{vert}(V)$ should include at most one of $male, female$ and at most one of $adopters$, $temp\_adopter$,non_adptr. *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 paper, 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 $adopters, temp\_adopter, non\_adptr$.

We now recapitulate the definition of generalized annotated logic programs from [Kif92b]. We assume the existence of a set AVar of variable symbols ranging over the unit real interval $[0, 1]$ and a set $\mathcal{F}$ of function symbols each of which has an associated arity. We start by defining annotations.

Definition 2.3 (annotation term)**.** (i) Any member of $[0, 1] \cup \mathsf{AVar}$ is an annotation.
(ii) If f is an n-ary function symbol over $[0, 1]$ and $t_1, \ldots, t_n$ are annotations, then so is $f(t_1, \ldots, t_n)$.

We define a separate logical language whose constants are members of $\mathbf{V}$ and whose predicate symbols consist of $\mathsf{VP} \cup \mathsf{EP}$. We also assume the existence of a set $\mathcal{V}$ of variable

[1] 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.

symbols ranging over the constants (vertices). No function symbols are present. Terms and atoms are defined in the usual way (cf. [Llo87]). 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.

Definition 2.4 (annotated atom/GAP-rule/GAP)**.** If A is an atom and μ is an annotation, then $A : \mu$ is an *annotated atom*. If $A_0 : \mu_0, A_1 : \mu_1, \ldots, A_n : \mu_n$ are annotated atoms, then

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

is called a *GAP rule*. When $n = 0$, the above GAP-rule is called a *fact*. A generalized annotated program Π is a finite set of GAP rules.

Every social network $SN = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ can be represented by the GAP $\Pi_{SN} = \{q(v) : 1 \leftarrow \mid v \in \mathbf{V} \wedge q \in \ell_{vert}(v)\} \cup \{ep(V_1, V_2) : w(V_1, V_2, ep) \leftarrow \mid (V_1, V_2) \in \mathbf{E} \wedge \ell_{edge}(V_1, V_2) = ep\}$.

Definition 2.5 (embedded social network)**.** A social network SN is said to be *embedded* in a GAP Π iff $\Pi_{SN} \subseteq \Pi$.

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

Example 2.6. The GAP Π_{cell} might consist of Π_{SN} using the social network of Figure 1 plus the GAP-rules:

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

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

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

Definition 2.7 (satisfaction/entailment)**.** An interpretation I *satisfies* a ground annotated atom $A : \mu$, denoted $I \models A : \mu$, iff $I(A) \geq \mu$. I satisfies the ground GAP-rule $AA_0 \leftarrow AA_1 \wedge \ldots \wedge AA_n$ (denoted $I \models AA_0 \leftarrow AA_1 \wedge \ldots \wedge AA_n$) 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 atom (rule) iff I satisfies all ground instances of it. GAP Π *entails* AA, denoted $\Pi \models AA$, iff every interpretation I that satisfies all rules in Π also satisfies AA.

As shown by [Kif92b], we can associate a fixpoint operator with any GAP Π that maps interpretations to interpretations.

Definition 2.8. Suppose Π is any GAP and I an interpretation. The mapping $\mathbf{T}_\Pi$ that maps interpretations to interpretations is defined as $\mathbf{T}_\Pi(I)(A) = \mathbf{sup}\{\mu \mid A : \mu \leftarrow AA_1 \wedge \ldots \wedge AA_n$ is a ground instance of a rule in Π and for all $1 \leq i \leq n$, $I \models AA_i\}$.

[Kif92b] show that $\mathbf{T}_\Pi$ is monotonic 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.

Thus, we see that any social network $\mathcal{S}$ can be represented as a GAP $\Pi_\mathcal{S}$. We will show (in Section 4) that many existing diffusion models of $\Pi_\mathcal{S}$ can be expressed as a GAP $\Pi \supseteq \Pi_\mathcal{S}$ by adding some GAP-rules describing the diffusion process to $\Pi_\mathcal{S}$.

3. Social Network Optimization (SNOP) Queries

In this section, we develop a formal syntax and semantics for optimization in social networks, taking advantage of the above embedding of SNs into GAPs. We see from queries **(Q1),(Q2)** that a SNOP-query looks for a set $\mathbf{V'}$ of vertices and has the following components: (i) an aggregate operator, (ii) an integer $k \geq 0$, (iii) a set of conditions that each vertex in $\mathbf{V'}$ must satisfy, and (iv) a *goal* atom $g(V)$ where g is a vertex predicate and V is a variable.

Aggregates. It is clear that in order to express queries like **(Q1),(Q2)**, we need aggregate operators which are mappings $agg : \mathsf{FM}([0,1]) \rightarrow \mathbb{R}$ ($\mathbb{R}$ is the set of 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 reals as input and return a single real.

Aggregates may be monotonic or not. We first define a partial ordering $\sqsubseteq$ on multi-sets of numbers as follows. $X_1 \sqsubseteq X_2$ iff there exists an *injective* mapping $\beta : X_1 \rightarrow X_2$ such that $(\forall x_1 \in X_1) x_1 \leq \beta(x_1)$. The aggregate agg is *monotonic (resp. anti-monotonic)* iff whenever $X_1 \sqsubseteq X_2$, it is the case that $agg(X_1) \leq agg(X_2)$ (resp. $agg(X_2) \leq agg(X_1)$).

Vertex condition. A vertex condition is a conjunction VC of annotated vertex atoms containing at most one variable.

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

Definition 3.1 (SNOP-query)**.** A *SNOP-query* is a 4-tuple $(agg, VC, k, g(V))$ where agg is an aggregate, VC is a vertex condition, $k \geq 0$ is an integer, and $g(V)$ is a goal atom.

If we return to our cell phone example, we can set $agg = \mathsf{SUM}$, $k = 3$ (for example), $VC = true$ and the goal to be $adopter(V)$. Here, the goal is to find a set X of annotated ground atoms of the form $adopter(v) : \mu$ such that X's cardinality is 3 or less and such that $\mathsf{SUM}\{\mu \mid adopter(v) : \mu \in X\}$ is maximized. Here, the SUM is applied to a multiset rather than a set.

Definition 3.2 (pre-answer/value)**.** Suppose an SN $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ is embedded in a GAP Π. A *pre-answer* to the SNOP query $Q = (agg, VC, k, g(V))$ w.r.t. Π is any set $\mathbf{V'} \subseteq \mathbf{V}$ such that: (i) $|\mathbf{V'}| \leq k$, (ii) for all vertices $v' \in \mathbf{V'}$, $lfp(\mathbf{T}_{\{\Pi \cup \{g(v'):1\leftarrow \mid v' \in \mathbf{V'}\}}) \models VC[V/v']$. We use $\mathsf{pre\_ans}(Q)$ to denote the set of all pre-answers to query Q.

The *value*, $value(\mathbf{V'})$, of a pre-answer $\mathbf{V'}$ is $agg(\{lfp(\mathbf{T}_{\Pi \cup \{g(v'):1\leftarrow \mid v' \in \mathbf{V'}\}})(g(V)) \mid V \in \mathbf{V}\})$ — here, 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 SNOP query. We say $value(I) = agg(\{I(g(v)) \mid v \in \mathbf{V}\})$.

If we return to our cell phone example, $\mathbf{V'}$ is the set of vertices to which the company is considering giving free plans. The value of this set ($value(\mathbf{V'})$) is computed as follows. Find the least fixpoint of $T_{\Pi'}$ where Π' is Π expanded with annotated atoms of the form $adopter(V') : 1$ for each vertex $V' \in \mathbf{V'}$. For each vertex $V \in \mathbf{V}$ (the entire set of vertices, not just $\mathbf{V'}$ now), we now find the confidence assigned by the least fixpoint. Summing up these confidences gives us a measure of the expected number of plan adoptees.

Definition 3.3 (answer)**.** Suppose an SN $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ is embedded in a GAP Π and $Q = (agg, VC, k, g(V))$ is a SNOP-query. A pre-answer $\mathbf{V'}$ is an *answer* to the SNOP-query Q iff the SNOP-query has no other pre-answer $\mathbf{V''}$ such that $value(\mathbf{V''}) > value(\mathbf{V'})$.[2]

The *answer set*, $\mathsf{ans}(Q)$, to the SNOP-query $Q = (agg, VC, k, g(V))$ w.r.t. Π is the set of all answers to Q.

Example 3.4. Consider the GAP Π_{cell} with the social network from Figure 1 embedded and the SNOP-query $Q_{cell} = (SUM, true, 3, will\_adopt)$. 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 yield a fixpoint where the vertex atoms formed with *will_adopt* in 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, and the corresponding set of vertices (where the vertex atom formed with *will_adopt* is annotated with 1) is $\{v_{15}, v_6, v_{12}, v_{18}, v_7, v_{10}\}$. 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 $value(\mathbf{V}'_2) = 6$. As $value(\mathbf{V}'_1) > value(\mathbf{V}'_2)$, it is easy to see that $\mathbf{V}'_1$ is an answer to this SNOP-query.

Theorem 3.5. *Answering SNOP-queries is NP-Hard.*[3]

Under some reasonable conditions, the problem of answering SNOP-queries is also in NP.

Theorem 3.6. *If both the aggregate function agg and the functions in $\mathcal{F}$ are polynomially computable, then the problem of finding an answer to a SNOP-query is in NP*[4]*.*

Most common aggregate functions like SUM, AVERAGE, Weighted average, MIN, MAX, COUNT are all polynomially computable. Moreover, the assumption that the functions in $\mathcal{F}$ are polynomially computable is also reasonable. The counting problem version of SNOP-query answering seeks to find the number of answers to a SNOP query. Unfortunately, this problem is $\#P$-complete under the same assumptions.

[2]Throughout this paper, we only treat maximization problems - minimizing an objective function f is the same as maximizing $-f$.

[3]**Proof Sketch:** Due to space constraints, we only explain the hardness result by reducing SET COVER to the problem of answering SNOP queries. Given a SET COVER problem instance consisting of a set S, a family $\mathcal{H} = \{H_1, \ldots, H_{max}\}$ of subsets of S, and a positive integer K, we can reduce this problem instance to a SNOP query by polynomially constructing a graph whose vertices correspond to the members of S and to the H_i's - there is an edge from an $s \in S$ to H_i iff $s \in H_i$. All edges have a weight of 1. Every vertex $v \in S$ has an associated propositional symbol *marked* set to "true." There is only one label and all edges are labeled with it and there are no integrity constraints. We have a GAP consisting of one rule $marked(v) : 1 \leftarrow marked(v') : 1 \wedge (v', v, label) : 1$. If we now consider the SNOP-query $(SUM, true, K, marked(v))$, we see that solutions to the SNOP-query (which cause certain H_i's to get marked) correspond precisely to a solution of the SET COVER problem.

[4]By abuse of notation, we refer to the obvious decision problem associated with answering SNOP-queries.

Theorem 3.7. *The counting version of the SNOP query answering problem is #P-complete.*

Although the counting version of the query is $\#P$-hard, finding the *union* of all answers to a SNOP query is no harder than a SNOP query. We shall refer to this problem as *SNOP-ALL* - and it reduces both to and from a regular SNOP query[5].

4. Applying SNOPs to Real Diffusion Problems

In this section, we briefly show how SNOPs may be used to solve two diffusion problems - one each in economics and disease spread.

The Jackson-Yariv Diffusion Model [Jac05]**.** In this framework, a set of agents is associated with each vertex in an undirected graph $\mathbf{G'} = (\mathbf{V'}, \mathbf{E'})$. Each agent has a default behavior (A) and a new behavior (B). Suppose d_i denotes the degree of a vertex v_i. [Jac05] use a function $g : \{0, \ldots, |\mathbf{V}| - 1\} \rightarrow [0, 1]$ to describe how the number of neighbors of v affects the benefits to v for adopting behavior B. For instance, $g(3)$ 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 c_i be the benefit and cost for vertex v_i to adopt behavior B, respectively. [Jac05] state that node v_i switches to behavior B iff $\frac{b_i}{c_i} \cdot g(d_i) \cdot \pi_i \geq 1$.

Returning to our cell-phone example, one could potentially use this model to describe the spread of the new plan. In this case, behavior B would be the use of the new plan. The associated SNOP-query would ask to simply find the nodes given a free plan that would maximize use of the plan in the network. Cost and benefit could be computed from factors such as income, time invested in switching plans, etc.

Given a Jackson-Yariv model consisting of $\mathbf{G'} = (\mathbf{V'}, \mathbf{E'})$ and g, we can set up an SN $(\mathbf{V'}, \mathbf{E''}, \ell_{vert}, \ell_{edge}, w)$ as follows. We define $\mathbf{E''} = \{(x, y), (y, x) \mid (x, y) \in \mathbf{E'}\}$. We have a single edge predicate symbol *edge* and ℓ_{edge} assigns 1 to all edges in $\mathbf{E''}$. Our associated GAP Π_{JY} now consists of Π_{SN} plus the single rule:

$$B(V_i) : \lfloor \frac{b_i}{c_i} \cdot g(\sum_j E_j) \cdot \frac{\sum_j X_j}{\sum_j E_j} \rfloor \leftarrow \bigwedge_{V_j | (V_j, V_i) \in \mathbf{E''}} (edge(V_j, V_i) : E_j \wedge B(V_j) : X_j)$$

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

The Kempe-Kleinberg-Tardos Framework.[Kem03] If we take the above construction, and for each v_i replace the $\frac{\sum_j X_j}{\sum_j E_j}$ in the head with a monotone *threshold function*, f_i, we have embedded the general framework of [Kem03], of which the [Jac05] model is a special case. It is important to note that the framework of [Kem03] captures a wide variety of diffusion models seen in social sciences and interacting particle systems. These include the "linear threshold model" - which is based on models in social science made popular by [Sch78] and [Gra78] and the "independent cascade model," introduced in [JG01]. However, this work provides a further generalization, as we allow for multiple properties to be "activated" on the vertices, permit labeled edges signifying different relationships, and provide a rule-based

[5]Our proofs of this statement rely on two constructions. First, a regular SNOP query, where the answer must be of size k, can be solved with k successive SNOP-ALL queries. Likewise, a SNOP-ALL query can be answered by solving $|\mathbf{V}|$ SNOP queries. Details are omitted due to lack of space.

framework which can allow for learned diffusion models. Additionally, [Kem03] does not solve SNOP queries with complex aggregates.

The SIR Model of Disease Spread. The SIR (*susceptible, infectious, removed*) model of disease spread [And79] 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* where 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. We would like to "find k vertices that would maximize the expected number of vertices that become infected". These are obviously good candidates to treat with appropriate medications.

Let $\mathcal{S} = (\mathbf{V}, \mathbf{E}, \ell_{vert}, \ell_{edge}, w)$ be an SN where each edge is labeled with the predicate symbol e and $w(v, v', e) = p_{v,v'}$. 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 $\mathcal{S}$ 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}:

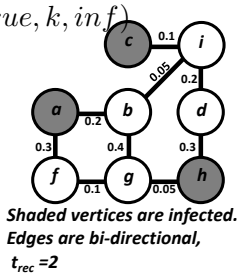
$$\begin{aligned} rec_i(V) : R &\leftarrow rec_{i-1}(V) : R \\ rec_1(V) : R &\leftarrow inf(V) : R \\ inf(V) : (1 - R) \cdot P_{V',V} \cdot (P_{V'} - R') &\leftarrow rec_{t_{rec}}(V) : R \wedge\ e(V', V) : P_{V',V} \ \wedge \\ &\quad\ inf(V') : P_{V'} \ \wedge\ rec_{t_{rec}}(V') : R'. \end{aligned}$$

The first rule says that if a vertex is in its $(i-1)$'th day of recovery with certainty R in the j'th iteration of the $\mathbf{T}_{\Pi_{SIR}}$ operator, then the vertex is i days into recovery (with the same certainty) in the $j+1$'th iteration of the operator. Likewise, second rule intuitively encodes the fact that if a vertex became infected with certainty 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 certainty. The last rule says that if a vertex V' has been infected with probability $P_{V'}$ and there is an edge from V' to V in the social network (weighted with probability $P_{V',V}$), and the vertex V' has recovered with certainty R', given the probability $1-R$ that V is not already recovered, (and hence, cannot be re-infected)[6], then the certainty that the vertex V gets infected is $(1 - R) \cdot P_{V',V} \cdot (P_{V'} - R')$. Here, $P_{V'} - R'$ is one way of measuring the certainty that V' has recovered (difference of the probability that it was infected and the probability it has recovered) and $P_{V',V}$ is the probability of infecting the neighbor.

To see how this GAP works, we execute a few iterations of the $\mathbf{T}_{\Pi_{SIR}}$ operator and show the fixpoint that it reaches on a toy sample graph shown in Figure 2. In this graph, the initial infected vertices are those shown in a shaded circle. The transmission probabilities weight the edges in the graph.

[6]Note that the SIS (Susceptible-Infectious-Susceptible) model [Het76], where an individual becomes susceptible to disease after recovering (as opposed to SIR, where an individual acquires immunity) can be easily represented by a modification to the described construction. Simply change the annotation function in the head of the third rule to $P_{V',V} \cdot (P_{V'} - R')$. In this way, we do not consider the probability that vertex V is immune.

The SNOP-query is $(SUM, true, k, inf)$ to count the number of infected vertices in the least fixpoint of T_Π. This query says "find the k vertices in the social network which, if infected, would cause the maximal number of vertices to become infected in the future." However, the above set of rules can be easily used to express other things. For instance, an epidemiologist may not be satisfied with only one set of k vertices that can cause the disease to spread to the maximum extent - as there may be another, disjoint set of k vertices that could cause the same effect. Note that a single set of vertices may still be sufficient for other applications, such as viral marketing. The epidemiologist may want to find all members of the population, that if in a group of size k could spread the disease to a maximum extent. This can be answered using a *SNOP-ALL* query, described in Section 3.



1	inf(a):1, inf(c):1, inf(d):1
2	rec_1(a):1, rec_1(c):1, rec_1(d):1, inf(b):0.2, inf(d):0.3, inf(f):0.3, inf(g):0.05, inf(i):0.1
3	rec_2(a):1, rec_2(c):1, rec_2(d):1, rec_1(b):0.2, rec_1(d):0.3, rec_1(f):0.3, rec_1(g):0.05, rec_1(i):0.1 inf(g):0.08
4	rec_2(b):0.2, rec_2(d):0.3, rec_2(f):0.3, rec_2(g):0.05, rec_2(i):0.1, rec_1(g):0.08
5	rec_2(g):0.08

Figure 2: Left: Sample network for disease spread. Right: annotated atoms entailed after each application of $\mathbf{T}_{\Pi_{SIR}}$ (maximum, non-zero annotations only).

5. Conclusion

In this paper, we described how General Annotated Logic Programs can be used to represent a variety of diffusion models in social networks. Based on this formulation, we presented the social network optimization problem (SNOP) - which queries the GAP for a set of nodes that cause a given phenomenon to spread through the social network to a maximum extent - shown here to be NP-Complete. We also showed how several well-known diffusion models can be represented in our framework. In future work, we intend to explore heuristic approaches for large sub-classes of SNOPs to answer queries on real-world datasets.

6. Acknowledgments

Some of the authors of this paper were funded in part by AFOSR grant FA95500610405, ARO grant W911NF0910206 and ONR grant N000140910685.

References

[And79] Roy M. Anderson and Robert M. May. Population biology of infectious diseases: Part i. *Nature*, 280(5721):361, 1979.

[Apt03] K. Apt. *Principles of constraint programming*. Cambridge University Press, 2003.

[Cow04] Robin Cowan and Nicolas Jonard. Network structure and the diffusion of knowledge. *Journal of Economic Dynamics and Control*, 28(8):1557 – 1575, 2004. doi:DOI:10.1016/j.jedc.2003.04.002. URL http://www.sciencedirect.com/science/article/B6V85-4B3K2R3-2/2/16e1c8fae6818c11bb45325c9b3ea4a1

[Dam99] C. Damasio, L. Pereira, and T. Swift. Coherent well-founded annotated logic programs. In *Proc. Intl. Conf. on Logic Programming and Non-Monotonic Reasoning*, pp. 262–276. Springer Lecture Notes in Computer Science Vol. 1730, 1999.

[FC08] H. Cruz F.C. Coelho, C. Codeco. Epigrass: A tool to study disease spread in complex networks. *Source Code for Biology and Medicin*, 3(3), 2008.

[Gra78] Mark Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978. doi:10.2307/2778111.
URL `http://dx.doi.org/10.2307/2778111`

[Het76] Herbert W. Hethcote. Qualitative analyses of communicable disease models. *Mathematical Biosciences*, 28(3-4):335 – 356, 1976. doi:DOI:10.1016/0025-5564(76)90132-2.
URL `http://www.sciencedirect.com/science/article/B6VHX-4771JSV-9/2/701b5ad988380270c29b4ab5dcd67bbe`

[Jac05] M. Jackson and L. Yariv. Diffusion on social networks. In *Economie Publique*, vol. 16, pp. 69–82. 2005.

[JG01] E. Muller J. Goldenberg, B. Libai. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3):211, 2001.

[Kem03] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 137–146. ACM, New York, NY, USA, 2003. doi:http://doi.acm.org/10.1145/956750.956769.

[Kif92a] M. Kifer and E. L. Lozinskii. A logic for reasoning with inconsistency. *J. Autom. Reasoning*, 9(2):179–215, 1992.

[Kif92b] Michael Kifer and V.S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. Log. Program.*, 12(3&4):335–367, 1992.

[Kra04] Stanislav Krajci, Rastislav Lencses, and Peter Vojts. A comparison of fuzzy and annotated logic programming. *Fuzzy Sets and Systems*, 144(1):173 – 192, 2004. doi:DOI:10.1016/j.fss.2003.10.019.
URL `http://www.sciencedirect.com/science/article/B6V05-49YH3XJ-2/2/1a631467fa197cb0a6f6ae93c1db1a59`

[Leo04] Nicola Leone, Francesco Scarcello, and V.S. Subrahmanian. Optimal models of disjunctive logic programs: Semantics, complexity, and computation. *IEEE Transactions on Knowledge and Data Engineering*, 16:487–503, 2004. doi:http://doi.ieeecomputersociety.org/10.1109/TKDE.2004.1269672.

[Llo87] John W. Lloyd. *Foundations of logic programming*. Springer-Verlag New York, Inc., 1987.

[Lu93] J.J. Lu, N.V. Murray, and E. Rosenthal. Signed formulas and annotated logics. In *Multiple-Valued Logic, 1993., Proceedings of The Twenty-Third International Symposium on*, pp. 48–53. 1993. doi:10.1109/ISMVL.1993.289582.

[Lu96] J. Lu. Logic programs with signs and annotations. *Journal of Logic and Computation*, 6(6):755–778, 1996.

[NE08] A. Pentland N. Eagle and D. Lazer. Mobile phone data for inferring social network structure. In *Proc. 2008 Intl. Conference on Social and Behavioral Computing*, pp. 79–88. Springer Verlag, 2008.

[Ryc08] Jan Rychtář and Brian Stadler. Evolutionary dynamics on small-world networks. *International Journal of Computational and Mathematical Sciences*, 2(1), 2008.
URL `www.waset.org`

[Sch78] Thomas C. Schelling. *Micromotives and Macrobehavior*. W.W. Norton and Co., 1978.

[Thi93] K. Thirunarayan and M. Kifer. A theory of nonmonotonic inheritance based on annotated logic. *Artificial Intelligence*, 60(1):23–50, 1993.

[Ven04] J. Venneksn, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *Proc. Intl. Conf. on Logic Programming*, pp. 431–445. Springer Lecture Notes in Computer Science Vol. 3132, 2004.

[Wat99] Duncan J. Watts. Networks, dynamics, and the small-world phenomenon. *The American Journal of Sociology*, 105(2):493–527, 1999.
URL `http://www.jstor.org/stable/2991086`