Generalized Communities in Networks

M. E. J. Newman and Tiago P. Peixoto

Phys. Rev. Lett. **115**, 088701 — Published 20 August 2015

DOI: 10.1103/PhysRevLett.115.088701

Generalized communities in networks

M. E. J. Newman[1,2] and Tiago P. Peixoto[3]

[1]*Department of Physics and Center for the Study of Complex Systems, University of Michigan, Ann Arbor, MI 48109, U.S.A.*
[2]*Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, U.S.A.*
[3]*Institut für Theoretische Physik, Universität Bremen, Hochschulring 18, D-28359 Bremen, Germany*

A substantial volume of research has been devoted to studies of community structure in networks, but communities are not the only possible form of large-scale network structure. Here we describe a broad extension of community structure that encompasses traditional communities but includes a wide range of generalized structural patterns as well. We describe a principled method for detecting this generalized structure in empirical network data and demonstrate with real-world examples how it can be used to learn new things about the shape and meaning of networks.

The detection and analysis of large-scale structure in networks has been the subject of a vigorous research effort in recent years, in part because of the highly successful application of ideas drawn from statistical physics [1, 2]. Particular energy has been devoted to the study of community structure, meaning the division of networks into densely connected subgroups, a common and revealing feature, especially in social and biological networks [3]. Community structure is, however, only one of many possibilities where real-world networks are concerned. In this paper, we describe a broad generalization of community structure that encompasses not only traditional communities but also overlapping or fuzzy communities, ranking or stratified structure, geometric networks, and a range of other structural types, yet is easily and flexibly detected using a fast, mathematically principled procedure which we describe. We give demonstrative applications of our approach to both computer-generated test networks and real-world examples.

Community structure can be thought of as a division of the nodes of a network into disjoint groups such that the probability of an edge is higher between nodes in the same group than between nodes in different groups. For instance, one can generate artificial networks with community structure using the stochastic block model, a mathematical model that follows exactly this principle. In the stochastic block model the nodes of a network are divided into k groups, with a node being assigned to group r with some probability γ_r for $r = 1 \ldots k$, and then edges are placed between node pairs independently with probabilities p_{rs} where r and s are the groups the nodes fall in. If the diagonal probabilities p_{rr} are larger than the off-diagonal ones, we get traditional community structure.

Alternatively, however, one can also look at the stochastic block model another way: imagine that we assign each node a random node parameter x between zero and one and edges are placed between node pairs with a probability $\omega(x, y)$ that is a function of the node parameters x and y of the pair. If $\omega(x, y)$ is piecewise constant with k^2 rectangular regions of size $\gamma_r\gamma_s$ and value p_{rs}, then this model is precisely equivalent to the traditional block model. But this prompts us to ask what is so special about piecewise constant functions? It is certainly possible that some networks might contain structure that is better captured by functions $\omega(x, y)$ of other forms. Why not let $\omega(x, y)$ take a more general functional form, thereby creating a generalized type of community structure that includes the traditional type as a subset but can also capture other structures as well? This is the fundamental idea behind the generalized structures of this paper: edge probabilities are arbitrary functions of continuous node parameters.

The idea is related to a number threads of work in the previous literature. One, in sociology and statistics, concerns "latent space" models, in which nodes in a network are located somewhere in a Euclidean space and are more likely to be connected if they are spatially close than if they are far apart [4]. Work within physics, mathematics, and computer science on graph metrics and embeddings addresses similar questions though with different methods [5–7]. The other thread, in the mathematics literature, concerns so-called "graphon" models and does not deal with the analysis of empirical data but with the mathematical properties of models, showing in particular that models of a kind similar to that described here are powerful enough to capture the properties of any theoretical ensemble of networks in the limit of large size, at least in the case where the networks are dense [8, 9].

In this paper, we define a specific model of generalized community structure and a method for fitting it to empirical data using Bayesian inference. The fit places each node of the network in the "latent space" of the node parameters x and, simultaneously, gives us an estimate of the probability function $\omega(x, y)$. Between them, these two outputs tell us a great deal about the structure a network possesses and the role each node plays within that structure. The method is computationally efficient in practice, allowing for its application to large networks, and provides significantly more insight than the traditional community division into discrete groups, or even recent generalizations to overlapping groups [10, 11].

We begin by defining a model that generates networks with the generalized community structure we are inter-

ested in. The model follows the lines sketched above, but with some crucial differences. We take n nodes and for each node u we generate a node parameter x_u uniformly at random in the interval $[0,1]$. Then between each pair of nodes u, v we place an undirected edge with probability

$$p_{uv} = \frac{d_u d_v}{2m}\,\omega(x_u, x_v), \tag{1}$$

where d_u, d_v are the degrees of the nodes, $m = \frac{1}{2}\sum_u d_u$ is the total number of edges in the network, and $\omega(x,y)$ is a function of our choosing, which we will call the *edge function*. Note that $\omega(x,y)$ must be symmetric with respect to its arguments for an undirected network such as this.

The inclusion of the degrees allows us to match the expected degree distribution of the model network to the distribution for the observed network [14]. Without it, the model effectively assumes a Poisson degree distribution, which is a poor fit to most networks [12, 13] and can cause the calculation to fail [15]. The factor $d_u d_v/2m$ is the probability of an edge between nodes with degrees d_u, d_v if edges are placed at random [2]. Hence $\omega(x_u, x_v)$ parametrizes the variation of the probability relative to this baseline level and is typically of order 1, making p_{uv} small in the limit where m becomes large.

Given the model, we fit it to empirical network data using the method of maximum likelihood. The probability or likelihood $P(\mathbf{A}, \mathbf{x}|\omega)$ that we generate a particular set of node parameters $\mathbf{x} = \{x_u\}$ and a particular network structure described by the adjacency matrix $\mathbf{A} = \{a_{uv}\}$ is

$$P(\mathbf{A}, \mathbf{x}|\omega) = \prod_{u<v} p_{uv}^{a_{uv}} (1-p_{uv})^{1-a_{uv}}. \tag{2}$$

(Recall that the node parameters x_u are chosen uniformly on the interval $[0,1]$, so their prior probability density is simply 1.) To find the value of the edge function $\omega(x,y)$ that best fits an observed network we want to maximize the marginal likelihood

$$P(\mathbf{A}|\omega) = \int P(\mathbf{A}, \mathbf{x}|\omega)\,\mathrm{d}^n\mathbf{x}, \tag{3}$$

or equivalently its logarithm, whose maximum falls in the same place. Direct maximization leads to a set of implicit equations that are hard to solve, even numerically, so instead we employ the following trick.

For any positive-definite function $f(x)$, Jensen's inequality says that

$$\log \int f(x)\,\mathrm{d}x \geq \int q(x) \log \frac{f(x)}{q(x)}\,\mathrm{d}x, \tag{4}$$

where $q(x)$ is any probability distribution over x such that $\int q(x)\,\mathrm{d}x = 1$. Applying (4) to the log of the marginal likelihood, Eq. (3), we get

$$\log \int P(\mathbf{A}, \mathbf{x}|\omega)\,\mathrm{d}^n\mathbf{x} \geq \int q(\mathbf{x}) \log \frac{P(\mathbf{A}, \mathbf{x}|\omega)}{q(\mathbf{x})}\,\mathrm{d}^n\mathbf{x}, \tag{5}$$

where $q(\mathbf{x})$ is any probability distribution over $\mathbf{x}$. It is straightforward to verify that the exact equality is recovered, and hence the right-hand side maximized, when

$$q(\mathbf{x}) = \frac{P(\mathbf{A}, \mathbf{x}|\omega)}{\int P(\mathbf{A}, \mathbf{x}|\omega)\,\mathrm{d}^n\mathbf{x}}. \tag{6}$$

Further maximization with respect to ω then gives us the maximum of the marginal likelihood, which is the result we are looking for. Put another way, a double maximization of the right-hand side of (5) with respect to both $q(\mathbf{x})$ and ω will achieve the desired result. And this double maximization can be conveniently achieved by alternately maximizing with respect to $q(\mathbf{x})$ using (6) and with respect to ω by differentiating.

This method, which is a standard one in statistics and machine learning, is called an expectation–maximization or EM algorithm [16]. It involves simply iterating these two operations from (for instance) a random initial condition until convergence. The converged value of the probability density $q(\mathbf{x})$ has a straightforward physical interpretation. Combining Eqs. (3) and (6), we have $q(\mathbf{x}) = P(\mathbf{A}, \mathbf{x}|\omega)/P(\mathbf{A}|\omega) = P(\mathbf{x}|\mathbf{A}, \omega)$. In other words, $q(\mathbf{x})$ is the posterior probability distribution on the node parameters $\mathbf{x}$ given the observed network and the edge function $\omega(x,y)$. It tells us the probability of any given assignment $\mathbf{x}$ of parameters to nodes. It is this quantity that will in fact be our primary object of interest here.

Substituting from Eqs. (1) and (2) into (5), keeping terms to leading order in small quantities and dropping overall constants, we can write the quantity to be maximized as

$$\iint_0^1 \sum_{uv} q_{uv}(x,y) \left[a_{uv} \log \omega(x,y) - \frac{d_u d_v \omega(x,y)}{2m} \right] \mathrm{d}x\,\mathrm{d}y, \tag{7}$$

where $q_{uv}(x,y) = \int q(\mathbf{x}) \delta(x_u - x) \delta(x_v - y)\,\mathrm{d}^n\mathbf{x}$ is the posterior marginal probability that nodes u, v have node parameters x, y respectively. The obvious next step is to maximize (7) by functional differentiation with respect to $\omega(x,y)$, but there is a problem. If we allow ω to take any form at all then it has an infinite number of degrees of freedom, which guarantees overfitting of the data. Put another way, physical intuition suggests that $\omega(x,y)$ should be smooth in some sense, and we need a way to impose that smoothness as a constraint on the optimization. There are a number of ways we could achieve this, but a common one is to express the function in terms of a finite set of basis functions. For nonnegative functions such as ω a convenient basis is the Bernstein polynomials of degree N:

$$B_k(x) = \binom{N}{k} x^k (1-x)^{N-k}, \qquad k = 0 \ldots N. \tag{8}$$

The Bernstein polynomials form a complete basis for polynomials of degree N and are nonnegative in $[0,1]$, so a linear combination $\sum_{k=0}^{N} c_k B_k(x)$ is also nonnegative provided $c_k \geq 0$ for all k. Our edge function $\omega(x,y)$ is a function of two variables, so we will write it as a double expansion in Bernstein polynomials

$$\omega(x,y) = \sum_{j,k=0}^{N} c_{jk} B_j(x) B_k(y), \tag{9}$$

which again is nonnegative for $c_{jk} \geq 0$. Bernstein polynomials have excellent stability properties under fluctuations of the values of the expansion coefficients, which makes them ideal for statistical applications such as ours. Note that since $\omega(x,y)$ is symmetric with respect to its arguments we must have $c_{jk} = c_{kj}$.

If $\omega(x,y)$ is constrained to take this form, then instead of the unconstrained maximization of (7) we now want to maximize with respect to the coefficients c_{jk}. To do this, we substitute from (9) into (7) and apply Jensen's inequality again, this time in its summation form $\log \sum_i f_i \geq \sum_i Q_i \log f_i/Q_i$. Then, by the same argument as previously, we find that the optimal coefficient values are given by the double maximization with respect to c_{jk} and $Q_{jk}(x,y)$ of

$$\iint_0^1 \mu(x,y) \sum_{jk} Q_{jk}(x,y) \log \frac{c_{jk} B_j(x) B_k(y)}{Q_{jk}(x,y)} \,\mathrm{d}x\,\mathrm{d}y - \iint_0^1 \nu(x)\nu(y) \sum_{jk} c_{jk} B_j(x) B_k(y) \,\mathrm{d}x\,\mathrm{d}y, \tag{10}$$

where

$$\mu(x,y) = \frac{1}{2m} \sum_{uv} a_{uv} q_{uv}(x,y), \quad \nu(x) = \frac{1}{2m} \sum_u d_u q_u(x), \tag{11}$$

and $q_u(x) = n^{-1} \sum_v \int q_{uv}(x,y)\,\mathrm{d}y$ is the marginal probability that node u has node parameter x. The maximization with respect to $Q_{jk}(x,y)$ is achieved by setting

$$Q_{jk}(x,y) = \frac{c_{jk} B_j(x) B_k(y)}{\sum_{jk} c_{jk} B_j(x) B_k(y)}, \tag{12}$$

and the maximization with respect to c_{jk} is achieved by differentiating, which gives

$$c_{jk} = \frac{\iint \mu(x,y) Q_{jk}(x,y) \,\mathrm{d}x\,\mathrm{d}y}{\int \nu(x) B_j(x) \,\mathrm{d}x \int \nu(y) B_k(y) \,\mathrm{d}y}. \tag{13}$$

Since all quantities on the right of this equation are nonnegative, $c_{jk} \geq 0$ for all j,k and hence $\omega(x,y) \geq 0$, as required.

The calculation of the optimal values of the c_{jk} is a matter of iterating Eqs. (12) and (13) to convergence, starting from the best current estimate of the coefficients. Note that the quantities μ and ν need be calculated only once each time around the EM algorithm, and both can be calculated in time linear in the size of the network in the common case of a sparse network with $m \propto n$. The integrals in Eq. (13) we perform numerically, using standard Gauss–Legendre quadrature.

This, in principle, describes a complete algorithm for fitting the model to observed network data, but in practice the procedure is cumbersome because of the denominator of Eq. (6), which involves an n-dimensional integral, where n is the number of nodes in the network, which is typically large. The traditional solution to this problem is to subsample the distribution $q(\mathbf{x})$ approximately using Monte Carlo importance sampling. Here, however, we use a different approach proposed recently by Decelle *et al.* [17], which employs belief propagation and returns good results while being markedly faster than Monte Carlo. The method focuses on a function $\eta_{u\to v}(x)$, called the belief, which represents the probability that node u has node parameter x if node v is removed from the network. The removal of node v allows us to write a self-consistent set of equations whose solution gives us the beliefs. The equations are a straightforward generalization to the present model of those given by Decelle *et al.*:

$$\eta_{u\to v}(x) = \frac{1}{Z_{u\to v}} \exp\biggl(-\sum_w d_u d_w \int_0^1 q_w(y)\omega(x,y)\,\mathrm{d}y\biggr) \times \prod_{\substack{w(\neq v)\\ a_{uw}=1}} \int_0^1 \eta_{w\to u}(y)\omega(x,y)\,\mathrm{d}y, \tag{14}$$

where $q_w(y)$ is again the marginal posterior probability for node w to have node parameter y and as before we have dropped terms beyond leading order in small quantities. The quantity $Z_{u\to v}$ is a normalizing constant which ensures that the beliefs integrate to unity:

$$Z_{u\to v} = \int_0^1 \exp\biggl(-\sum_w d_u d_w \int_0^1 q_w(y)\omega(x,y)\,\mathrm{d}y\biggr) \times \prod_{\substack{w(\neq v)\\ a_{uw}=1}} \int_0^1 \eta_{w\to u}(y)\omega(x,y)\,\mathrm{d}y\,\mathrm{d}x. \tag{15}$$

The belief propagation method consists of the iteration of these equations to convergence starting from a suitable initial condition (normally the current best estimate of the beliefs). The equations are exact on networks that take the form of trees, or on locally tree-like networks in the limit of large network size (where local neighborhoods of arbitrary size are trees). On other networks, they are approximate only, but in practice give excellent results.

Once we have the values of the beliefs, the crucial two-node marginal probability $q_{uv}(x,y)$ is given by

$$q_{uv}(x,y) = \frac{\eta_{u\to v}(x)\eta_{v\to u}(y)\omega(x,y)}{\iint_0^1 \eta_{u\to v}(x)\eta_{v\to u}(y)\omega(x,y)\,\mathrm{d}x\,\mathrm{d}y}. \tag{16}$$

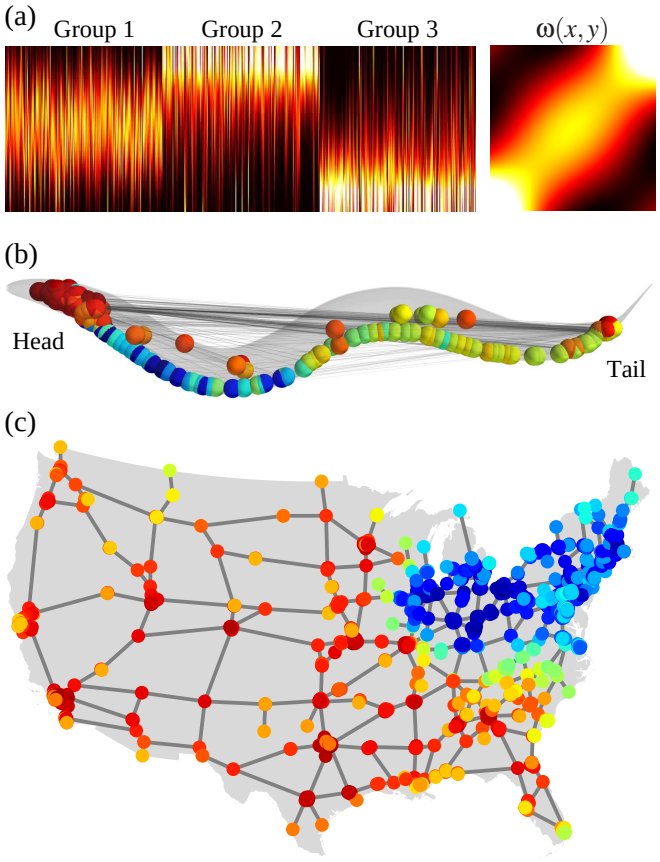

FIG. 1: (a) Left: density plot of the posterior marginal probability densities $q_u(x)$ that node u has node parameter x for an application of our algorithm to a 600-node stochastic block model with three groups. Colors indicate the probabilities and there are 600 columns, one for each node. Right: density plot of the edge function $\omega(x, y)$. (b) The neural network of the worm *C. elegans*, drawn in real space, as it falls within the body of the worm. Colors represent the average values of the node parameters x_u inferred for each neuron by our algorithm. (c) Network representation of the interstate highways of the contiguous United States. Again, node colors represent the average node parameters x_u.

Armed with these quantities for every node pair connected by an edge, we can evaluate $\mu(x, y)$ and $\nu(x)$ from Eq. (11) then iterate Eqs. (12) and (13) to compute new values of the parameters c_{jk}, and repeat. The final algorithm is efficient, with each iteration of the belief propagation equations running in time linear in the network size [14].

We give three example applications of our methods, one to a computer-generated benchmark network and the others to real-world networks displaying nontrivial latent-space structure that is readily uncovered by our algorithm.

For our first example, we use a computer-generated test network created using the standard stochastic block model, with $n = 600$ nodes divided into three equally sized groups of 200 nodes each, with probabilities $p_{\rm in} = c_{\rm in}/n$ and $p_{\rm out} = c_{\rm out}/n$ for edges between nodes in the same and different groups respectively and $c_{\rm in} = 15$, $c_{\rm out} = 3$. Figure 1a shows a density plot of the marginal probability distributions $q_u(x)$ on the node parameters calculated by our algorithm using a degree-4 (quartic) polynomial representation of the edge function ω. (We also used quartic representations for the other examples below.) The plot consists of 600 columns, one for each node, color coded to show the value of $q_u(x)$ for the corresponding node. As the plot shows, the algorithm has found the three known groups in the network, placing them at three widely spaced points in the latent space of the node parameters. (In this case, the first group is placed in the middle, the second at the top, and the third at the bottom, but all orders are equivalent.) We also show a plot of the inferred edge function $\omega(x, y)$, which in this case has a heavy band along the diagonal, indicating "assortative" structure, in which nodes are primarily connected to others in the same group.

Our second example is a real-world network, the neural network of the nematode (roundworm) *C. elegans*, which has been mapped in its entirety using electron microscopy [18, 19] and contains a total of 299 neurons. The worm has a long tubular body, with neurons arranged not just in its head but along its entire length. Neurons tend to be connected to others near to them, so we expect spatial position to play the role of a latent variable and our algorithm should be able to infer the positions of neurons by examining the structure of network. Figure 1b shows that indeed this is the case. The figure shows the network as it appears within the body of the worm, with nodes colored according to the mean values of the node parameters found by the algorithm, and we can see a strong correlation between node color and position. The largest number of nodes is concentrated in the head, mostly colored red in the figure; others along the body appear in blue and green. If we did not know the physical positions of the nodes in this case, or if we did not know the correlation between position and network structure, we could discover it using this analysis.

Our third example, shown in Fig. 1c, is an analysis of the network of interstate highways in the contiguous United States. This network is embedded in geometric space, the surface of the Earth. Again we would expect our algorithm to find this embedding and indeed it does. The colors of the nodes represent the mean values of their node parameters and there is a clear correspondence between node color and position, with the inferred node parameters being lowest in the north-east of the country and increasing in all directions from there. (Note that even though the portion of the network colored in red and orange appears much larger than the rest, it is in fact about the same size in terms of number of nodes because of the higher density of nodes in the north-east.) The true underlying space in this case has two dimen-

sions, where our model has only one, and this suggests a potential generalization to latent spaces with two (or more) dimensions. It turns out that such a generalization is possible and straightforward, but we leave the developments for future work.

To summarize, we have in this paper described a generalized form of community structure in networks in which network nodes are placed at positions in a continuous space and edge probabilities depend in a general manner on those positions. We have given a computationally efficient algorithm for inferring such structure from empirical network data, based on a combination of an EM algorithm and belief propagation, and find that it successfully uncovers nontrivial structural information about both artificial and real networks in example applications.

The authors thank Cris Moore and Cosma Shalizi for useful discussions. This research was funded in part by the US National Science Foundation under grants DMS–1107796 and DMS–1407207 and by the University of Bremen under funding program ZF04.

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, Physics Reports **424**, 175 (2006).
[2] M. E. J. Newman, *Networks: An Introduction* (Oxford University Press, Oxford, 2010).
[3] S. Fortunato, Phys. Rep. **486**, 75 (2010).
[4] P. D. Hoff, A. E. Raferty, and M. S. Handcock, J. Amer. Stat. Assoc. **97**, 1090 (2002).
[5] F. R. K. Chung, *Spectral Graph Theory*, no. 92 in CBMS Regional Conference Series in Mathematics (American Mathematical Society, Providence, RI, 1997).
[6] M. Belkin and P. Niyogi, Neural Computation **15**, 1373 (2003).
[7] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá, Phys. Rev. E **82**, 036106 (2010).
[8] L. Lovász, *Large Networks and Graph Limits*, vol. 60 of *American Mathematical Society Colloquium Publications* (American Mathematical Society, Providence, RI, 2012).
[9] C. Borgs, H. C. Jennifer T. Chayes and, and Y. Zhao, Preprint arxiv:1401.2906 (2014).
[10] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, Nature **435**, 814 (2005).
[11] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, Journal of Machine Learning Research **9**, 1981 (2008).
[12] A.-L. Barabási and R. Albert, Science **286**, 509 (1999).
[13] L. A. N. Amaral, A. Scala, M. Barthélemy, and H. E. Stanley, Proc. Natl. Acad. Sci. USA **97**, 11149 (2000).
[14] See Supplemental Material at [URL will be inserted by publisher] for a discussion of the degree-corrected model and details of algorithm implementation.
[15] B. Karrer and M. E. J. Newman, Phys. Rev. E **83**, 016107 (2011).
[16] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions* (Wiley-Interscience, New York, 2008), 2nd ed.
[17] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. Lett. **107**, 065701 (2011).
[18] J. G. White, E. Southgate, J. N. Thompson, and S. Brenner, Phil. Trans. R. Soc. London **314**, 1 (1986).
[19] B. Szigeti, P. Gleeson, M. Vella, S. Khayrulin, A. Palyanov, J. Hokanson, M. Currie, M. Cantarelli, G. Idili, and S. Larson, Frontiers in Computational Neuroscience **8**, 137 (2014).
[20] See Supplemental Material at [URL will be inserted by publisher], which includes Refs. [21] and [22].
[21] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, Phys. Rev. E **84**, 066106 (2011).
[22] This research used data from Add Health, a program project directed by Kathleen Mullan Harris and designed by J. Richard Udry, Peter S. Bearman, and Kathleen Mullan Harris at the University of North Carolina at Chapel Hill, and funded by grant P01-HD31921 from the Eunice Kennedy Shriver National Institute of Child Health and Human Development, with cooperative funding from 23 other federal agencies and foundations. Special acknowledgment is due Ronald R. Rindfuss and Barbara Entwisle for assistance in the original design. Information on how to obtain the Add Health data files is available on the Add Health website (http://www.cpc.unc.edu/addhealth). No direct support was received from grant P01-HD31921 for this analysis.