
Computational & Mathematical Organization Theory, 11, 291–305, 2005

Some Simple Algorithms for Structural Comparison

CARTER T. BUTTS
Department of Sociology and Institute for Mathematical Behavioral Sciences, SSPA 2145, University of California-Irvine, Irvine, CA 92697
email: buttsc@uci.edu

KATHLEEN M. CARLEY
Institute for Software Research International, Center for the Computational Analysis of Social and Organizational Systems, and H.J. Heinz III School of Public Policy and Management, Carnegie Mellon University
email: kcarley@ece.cmu.edu

Abstract

Structural comparison (i.e., the simultaneous analysis of multiple structures) is a problem which arises frequently in such diverse arenas as the study of organizational forms, social network analysis, and automated text analysis. Prior work has demonstrated the applicability of a range of standard multivariate analysis procedures to the structural comparison problem. Here, some simple algorithms are provided which elucidate several of these methods in an easily implemented form.

Keywords: multivariate methods, structural comparison, graph labeling, algorithms

Numerous questions in both applied and basic social science research deal with the problem of structural comparison; that is, the examination of similarities and differences among relational structures of one sort or another. The specific content of these structures may vary greatly by problem domain—with examples ranging from team structures and task assignment networks to mental models and networks of acquaintance or friendship—but where a graphical representation may be employed for the structures in question, we may map particular structural comparison problems to a more general one of analyzing collections of networks. In a prior paper, Butts and Carley (2001) present a general framework for the analysis of graph sets (i.e., sets of networks) by means of edge set comparison. While several applications of this framework to distance and covariance-based methods are demonstrated, algorithmic details are not included. Here, we build on this previous work by providing algorithms to implement the two basic graph comparison measures—structural distances and covariances—described therein. These algorithms are presented in a generic, pseudo-code form which should be easily adapted to most modern programming languages.[1]

It should be noted that the algorithms presented here have been written with an emphasis on simplicity, rather than efficiency; their purpose is to provide a further elaboration of the methods described in Butts and Carley (2001), rather than to attempt an optimal implementation. That said, the procedures given here should be of sufficient efficiency to be of use to an interested reader for typical social network analysis tasks. As noted below, some extremely standard but nontrivial procedures (e.g., random number generation) have been

employed here without further explanation. The reader is advised to consult a standard numerical methods text e.g., Press et al. (1992), and Gentle (1998) for additional information on these procedures.

1. Notation and External Routines

Before proceeding to the algorithms themselves, we begin by setting forth some basic notation which will be used throughout this paper. We will also identify certain external routines which we shall employ from time to time (but which are not provided here), and will provide a brief description of the common labeling problem. Although a thorough discussion of this last problem is beyond the scope of this paper, a limited introduction is necessary to illuminate the algorithms which follow.

1.1. Some Notational Conveniences

Throughout this document, we will utilize the following notation. Vectors are generally represented as lower-case letters in boldface (e.g., $\mathbf{v}$), with matrices being shown in upper-case (e.g., $\mathbf{D}$). Typically, subscripts are used to identify particular objects, with parenthetical superscripts indicating element selection. (Thus, $\mathbf{D}_i^{(jk)}$ is the j,kth element of matrix $\mathbf{D}_i$.) $G = (V, E)$ denotes the graph on vertex set V with edge set E; we also employ the notation $V(G)$ and $E(G)$ (respectively) to denote these sets in relation to a given graph. For convenience, the order of G (i.e. the graph size, $|V(G)|$) is denoted by n. Often, we will refer to G in terms of its adjacency matrix, $\mathbf{A}$, which will be identified in the same manner (i.e., subscript) as G where more than one graph is present.

When setting the value of a variable within the context of an algorithm, the expression $a := b$ should be understood as setting the value of a to the value of b. Computational complexity of algorithms is given in standard "big-O" notation, with $\mathcal{O}(f(x))$ read as "order $f(x)$." κ and ℓ refer to special functions, which will be described below; as will be noted, these are also sometimes treated as vectors, since they can be expressed in this form. Finally, the function **vec** coerces matrices to vectors by row concatenation. Other notation used here will be explained as necessary.

1.2. Assumed External Routines

In addition to basic control structures (e.g., if/then/else constructs, for and repeat loops) and arithmatic operations, we shall refer from time to time to two routines which are non-trivial, but which are both standardized and low-level enough that their presentation is superfluous here. These are generation of uniform deviates on the continuous (a, b) interval (denoted $\texttt{urand}(a, b)$) and generation of uniform deviates on the discrete $[a, b]$ interval (denoted $\texttt{irand}(a, b)$). Elementary algorithms for these routines may be found in Press et al. (1992) and Gentle (1998).

Another basic tool which will be used throughout this document is simulated annealing (Metropolis et al., 1953; Kirkpatrick et al., 1983), a simple family of algorithms for

heuristic optimization. Although the annealing algorithms are discussed where employed, the treatment here is relatively cursory. More in-depth discussion of annealing methods and their properties can be found in Otten and van Ginneken (1989).

1.3. Some Comments on Labeling

The general approach to graph comparison set forth by Butts and Carley (2001) can be thought of as deriving from three elements: a mapping of multiple graph vertex sets onto a common "comparison" set; a choice of dyadic similarity/distance measure between edge sets (conditional on the common vertex set); and a secondary procedure for analyzing the set of dyadic similarities/distances. Of these problems, the first—referred to here generically as the "labeling problem"—is the least studied. Because it is critical to the procedures which follow, we shall provide a very brief description of the problem before continuing on to the algorithms themselves.

Consider two graphs, G_1 and G_2, and assume that $|V(G_1)| = |V(G_2)|$.[2] To compare the elements of $E(G_1)$ and $E(G_2)$, we must possess a mapping which takes one set onto the other; since each edge is uniquely identified with an ordered pair of vertices, we can in turn transform this problem into one of establishing a one-to-one mapping between $V(G_1)$ and $V(G_2)$. In practice, we accomplish this by means of two functions: a *labeling function*, ℓ, which assigns (local) labels to vertices; and a *coloring function*, κ, which takes vertex labels into a set of shared labels (here called "colors") which are common across the graphs being examined. Intuitively, these shared labels identify which sets of positions may be taken as equivalent to one another for purposes of comparison; positions may be exchanged iff they are of the same color. Without loss of generality, we take the ordering indices $(1, \ldots, n)$ of vertices as labels, and we further assume that vertices are initially labeled (ordered) in some arbitrary fashion (as must be the case for the adjacency matrix to exist).

Our notation for κ and ℓ is somewhat polymorphic, and operates as follows. As one would expect, for $i \in (1, \ldots, n)$, $\ell : i \rightarrow (1, \ldots, n)$; that is, given a vertex position (here synonymous with the vertex itself), ℓ takes said position into a new position. By assumption, ℓ is constrained to be one-to-one on $(1, \ldots, n)$ (and hence a permutation vector for $\mathbf{A}$). $\ell^{(i)}$ is the ith value of ℓ, or $\ell(i)$, and we use the former notation when *setting* the values of ℓ.[3] (Thus, $\ell^{(3)} := 5$ maps the third vertex position onto the fifth vertex position.) When given sets of vertices/positions, it is further understood that ℓ returns the correspondingly relabeled versions of each. This allows us to apply the labeling function to graphs: $\ell(G) = (\ell(V), E)$ is the graph formed by relabeling (i.e., permuting) the vertices of G via labeling function ℓ.

The coloring function, κ, behaves similarly in most respects, save in that the return values of κ are not vertex positions (and hence $\kappa(V)$ is not a permutation vector). Instead, κ takes positions into a countable set of "colors" having cardinality $\leq n$. Given two graphs G_1 and G_2 with coloring functions κ_1 and κ_2, the edge sets of G_1 and G_2 are comparable for any labeling pair ℓ_1, ℓ_2 such that $\kappa_1(\ell_1(V(G_1))) = \kappa_2(\ell_2(V(G_2)))$. Such a pair of labelings is said to be "accessible" or "common" for the graph pair and associated coloring functions.[4] Without loss of generality, we may fix one labeling function to the default labeling, which results in the simplified constraint $\kappa_1(\ell(V(G_1))) = \kappa_2(V(G_2))$.

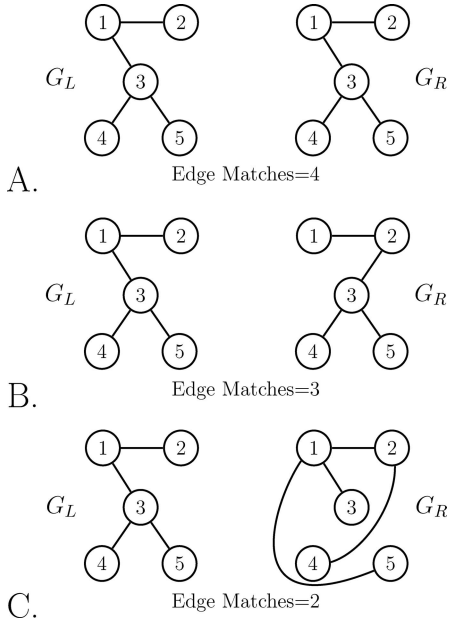

Figure 1. Effect of relabeling on edge matches.

In general, there may exist many ℓ which satisfy this constraint, and hence comparison of the edge sets $E(G_1)$, $E(G_2)$ will typically require us to consider multiple possibilities. The practical consequences of such diversity are illustrated by the simple example of figure 1. Panel A depicts two simple graphs (G_L and G_R), as might arise from observed communication ties within two different work groups. A very simple measure of similarity for the two structures is the number of matching edges between them; i.e., the number of i, j pairs such that $\{i, j\} \in E(G_L) \cap E(G_R)$. As shown, the graphs of panel A are isomorphic, with four matching edges. Now, let us presume that an accessible labeling allows us to exchange positions 1 and 2 within G_R. As panel B shows, the resulting structures now share only three edges. An even more drastic relabeling is depicted in panel C, leading to a total of two edge matches. Clearly, then, a naive attempt to compare G_L and G_R using something like an edge match statistic may yield different results depending on which labeling happens to be employed. Thus, a major task of the edge set comparison programme is the development of ways of coping with common labelings which are not unique.

Non-uniqueness of common labelings can arise, in practice, for a variety of reasons. The simplest case occurs when comparisons are sought between the underlying structures of two or more graphs, irrespective of vertex identities. For example, consider the problem of comparing the communication structures of two workgroups with non-overlapping membership. In this case, the graphs are properly unlabeled with respect to one another—there is no particular reason to identify an individual in one group with any specific individual in the other group—and any labeling of the respective vertices is hence accessible. Such

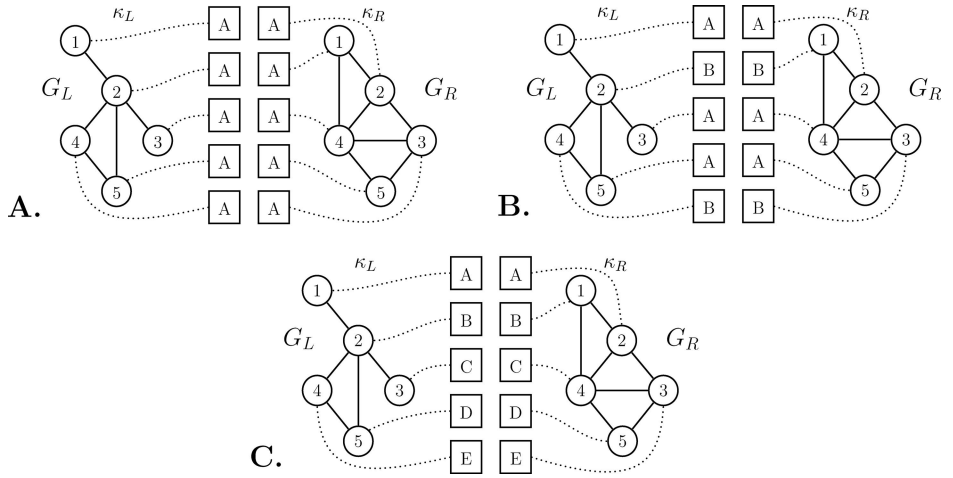

Figure 2. Sample graphs and coloring functions.

a scenario is illustrated in panel A of figure 2. Here, we have two graphs, G_L and G_R, whose initial vertex labels are shown with superimposed numerical values. The respective coloring functions κ_L and κ_R, which take positions into colors, are represented via dotted lines connecting positions with lettered boxes (each representing a particular color). In the case of panel A, all positions for each graph are mapped to a single color (denoted "A"); in the above notation, $\kappa_L(V(G_L)) = \kappa_R(V(G_R)) = (A, A, A, A, A)$. Note that, for any labeling function ℓ, we also have $\kappa_L(\ell(V(G_L))) = (A, A, A, A, A)$ (and likewise for the right-hand graph). Since all positions are mapped to the same color, relabeling either graph will have no effect on the shared labels. Formally, $\kappa_L(\ell(V(G_L))) = \kappa_R(\ell'(V(G_R)))$ for all ℓ, ℓ', and hence all labeling functions must be considered when comparing G_L and G_R.

A less extreme version of this problem may arise when vertices belong to distinct, identifiable classes, but where individuals are otherwise exchangeable across structures. In comparing organizational reporting structures, for instance, one might consider management distinct from staff without being able to link particular personnel across firms. In this case, the corresponding graphs will be partially labeled, and multiple common labelings will exist. A simple example of this kind is shown in panel B of figure 2. Here, κ_L and κ_R map their respective graphs (G_L and G_R) onto two colors. These colors (denoted "A" and "B") represent nonexchangeable classes of vertices; in the above example, colors A and B might refer to "staff" and "management," respectively. From the figure, we can see that $\kappa_L(V(G_L)) = (A, B, A, B, A)$ and $\kappa_R(V(G_R)) = (B, A, B, A, A)$. Since $\kappa_L(V(G_L)) \neq \kappa_R(V(G_R))$, it follows that the initial labeling is not accessible. However, the G_L labelings (2, 1, 4, 3, 5) and (4, 1, 2, 5, 3) *are* accessible, as is any other labeling ℓ such that $\kappa_L(V(\ell(G_L))) = \kappa_R(V(G_R))$. In the context of the reporting structure problem, the accessible labelings are those which match managers with managers and staff with staff. The set of such labelings may be large, but it is generally much smaller than the set of all labelings (e.g., as in panel A of figure 2).

Finally, the limiting case of non-exchangeability exists when each vertex in a given graph can be mapped to exactly one vertex in the graph to be compared. In contrast with the "unlabeled" and "partially labeled" comparisons considered above, the graphs in this case are said to be "fully labeled" with respect to one another. Situations in which one seeks to compare two relations on a single group of individuals (e.g., friendship and advice seeking behavior within a workgroup) are typical examples. Panel C of figure 2 illustrates such a scenario, with κ_L and κ_R mapping the respective vertices of G_L and G_R onto the colors "A," "B," "C," "D," and "E." As with panel B, the default labeling is not part of the accessible set. In fact, holding G_R fixed, the only G_L labeling which is accessible is $\ell = (2, 1, 4, 3, 5)$. In most practical settings, such graphs are coded such that the default labeling is accessible—thus, one generally drops such notation in the fully labeled case.

While the above serve to illustrate some simple cases, a detailed consideration of the choice of coloring functions is beyond the scope of this paper. Rather, we are here concerned with the computational problem of computing edge comparison statistics (specifically, structural distances and covariances) under nondegenerate sets of accessible labelings. Following Butts and Carley (2001), our approach is analogous to optimal matching (Abbott and Tsay, 2000) on the set of accessible labelings: given the graphs and associated coloring functions, we attempt to compute the minimum distance/maximum covariance attainable under the accessibility criterion. It is to this problem, then, that we now turn.

2. Calculating Structural Distances

Distance methods have long been a fixture of exploratory multivariate analysis, particularly where complex entities are involved. The use of distance-based approaches for graph comparison follows Banks and Carley (1994), with extensions by Butts (1998), and Butts and Carley (2001) to the unlabeled and partially labeled cases. Here, we show how the distances between structures may be calculated for graphs under arbitrary labeling assumptions. Distance matrices constructed using these methods can be used for a wide array of purposes, including multidimensional scaling (Torgerson, 1952; Borg and Lingoes, 1987) and cluster analysis (Romesburg, 1984).

The family of distances considered by Butts and Carley (2001) is based on various transformations of the summed differences between adjacency matrices. The Hamming distance (Hamming, 1950), for instance, can be thought of as the absolute difference between the dichotomized adjacency matrices, and transformations on the undichotomized absolute differences (i.e., $(\sum_i \sum_j |\mathbf{A}_1^{(ij)} - \mathbf{A}_2^{(ij)}|^\gamma)^{\frac{1}{\gamma}}$) yields a simple family of Minkowski metrics. Here, we consider only the absolute difference, since calculation of related metrics follows quite naturally from this simple algorithm.

The procedure for calculating the distance in absolute differences between two graphs, G_1 and G_2, is shown in Algorithm 1. Note that we assume that both graphs have been mapped to the same vertex set; that is, we assume that there is a one-to-one mapping between the vertices of G_1 and those of G_2, a mapping which is implicit in the order of rows and columns in their respective adjacency matrices (which must be of identical dimension). The nature of this assumption is discussed in depth elsewhere (Butts and Carley, 2001)—and we will

Algorithm 1: Graph Distance

```
1: procedure gdist(G_1, G_2)
2: d := 0
3: for i ∈ (1, ..., n) do
4:    for j ∈ (1, ..., n) do
5:       d := d + |A_1^(ij) − A_2^(ij)|
6:    end for
7: end for
8: return d
```

not belabor it here—but some special considerations which bear upon the calculation of distances will be described presently.

The time complexity of `gdist` is obviously $\mathcal{O}(n^2)$; total execution time may be reduced by a factor of two in the undirected case by summing only the upper or lower triangles of the respective adjacency matrices. This simple algorithm obviously suffices in the special case of graph pairs with a unique common labeling. In the more general case—where the common labeling is not unique—we must adopt a somewhat different procedure. In particular, we seek in the non-unique case to identify the *minimum* distance associated with the set of common labelings. This distance (whatever the underlying metric) is called the *structural distance*, and it is trivial to see that the graph distance dealt with previously is a special case of the structural distance when the set of common labelings is degenerate.

Calculation of the structural distance involves searching the space of common labelings for a minimum distance, a problem whose worse-case complexity scales with $\mathcal{O}(n!)$; in practice, then, heuristics must generally be employed to estimate the true structural distance. An approach to this problem using canonical labeling was investigated by Butts and Carley (1998), and Butts (1998) employed genetic algorithms to minimize structural distance, but both approaches proved relatively cumbersome for large graphs. Subsequent experimentation has suggested that simulated annealing methods work well for the distance minimization problem, and such methods have the additional virtue of being easy to implement; it is such a method which we illustrate here.

On order to find the permutation which minimizes the structural distance, it is obviously necessary to compare distances induced by various labelings; while this can be accomplished by multiple calls to `gdist`, such calls are computationally expensive for large graphs. Where the labelings in question differ only slightly, invocation of `gdist` is unnecessary. In particular, for a labeling change involving only single dyadic permutation, it is possible to compute the graph distance change in linear time. Formally, let us consider graphs G_1, G_2, and let ℓ be a labeling which exchanges vertices a and b. Then the quantity $\texttt{gdist}(G_1, \ell(G_2)) - \texttt{gdist}(G_1, G_2)$ can be computed by the function `gddelta` shown in Algorithm 2 with $\mathcal{O}(n)$ time complexity. This is a marked improvement over the naive approach, and we exploit it in our structural distance computations.

We are now ready to compute structural distances. The `sdist` procedure (shown as Algorithm 3) takes as inputs two graphs and their respective coloring functions; it returns

Algorithm 2: Change in Graph Distance Under Single Dyad Permutation

1: **procedure** gddelta(G_1, G_2, a, b)
2: $d := 0$
3: **for** $i \in (1, \ldots, n) \setminus \{a, b\}$ **do**
4: $d := d + (|A_1^{(ai)} - A_2^{(bi)}| - |A_1^{(ai)} - A_2^{(ai)}|) + (|A_1^{(bi)} - A_2^{(ai)}| - |A_1^{(bi)} - A_2^{(bi)}|)$
$+ (|A_1^{(ia)} - A_2^{(ib)}| - |A_1^{(ia)} - A_2^{(ia)}|) + (|A_1^{(ib)} - A_2^{(ia)}| - |A_1^{(ib)} - A_2^{(ib)}|)$
5: **end for**
6: $d := d + (|A_1^{(aa)} - A_2^{(bb)}| - |A_1^{(aa)} - A_2^{(aa)}|) + (|A_1^{(bb)} - A_2^{(aa)}| - |A_1^{(bb)} - A_2^{(bb)}|)$
$+ (|A_1^{(ab)} - A_2^{(ba)}| - |A_1^{(ab)} - A_2^{(ab)}|) + (|A_1^{(ba)} - A_2^{(ab)}| - |A_1^{(ba)} - A_2^{(ba)}|)$
7: **return** d

the structural distance (using the metric of gdist) and, as a byproduct, a labeling which produces this distance. (Without loss of generality, we may hold the labeling of one graph constant, and relabel the other to find the minimum distance.) In addition to these inputs, the procedure requires three parameters: $t_0 > 0$, the "initial temperature" for the algorithm; $t_f > 0$, the "freezing point"; and $\gamma \in (0, 1)$, the "cooling rate." Within the annealer, the "temperature" is a parameter which scales the effect of candidate deficiency on the probability of acceptance. Specifically, candidate moves are accepted with probability $\exp(-d_\Delta/t)$, where $-d_\Delta$ is the difference between the "old" distance (d_o) and the candidate distance under ℓ_c, and t is the temperature. Thus, when $t \gg 1$, a move which increases the estimated distance is more likely to be accepted; as $t \to 0$, the probability of such a move similarly approaches 0. Initially, $t = t_0$, but the temperature is reduced by a factor of γ with each iteration until the freezing point t_f is achieved (at which point the algorithm terminates).

Choice of optimal temperature ranges and cooling points requires some experimentation, as the system should cool and freeze quickly enough to avoid inefficiency while not cooling/freezing so rapidly as to converge prematurely to a (deficient) local optimum. Tracking the progress of the algorithm (d_b and d_o) as well as the rejection probabilities can provide some sense of whether the annealer has been well-tuned. In particular, one should verify that the algorithm is free to move (i.e., acceptance probabilities are high) in its early iterations, that improvements in the solution have not been obtained for several iterations before the freezing point, and that the last move (d_o) is at or close to the final solution (d_b). Violations of one or more of these conditions suggest that the annealer is either failing to explore the search space adequately, or else is failing to converge, and its parameters should be adjusted accordingly.

The label drawing subroutine (lines 8–15) given here is straightforward, and reasonably efficient where the coloring classes of κ are large (e.g., as one approaches the unlabeled case). Where the opposite holds (e.g., as one approaches the uniquely labeled limit), the number of iterations required for each execution of the label drawing subroutine approaches a geometric distribution with a parameter of approximately $(\frac{n}{2})^{-1}$; the expected number of executions, then, grows with $\mathcal{O}(n^2)$ in the limit. Although this can certainly be improved upon (e.g., by building a list of permissible exchanges and drawing only from within those),

Algorithm 3: Structural Distance

```
 1: procedure sdist(G_1, G_2, κ_1, κ_2)
 2: ℓ_o := (1, ..., n)
 3: d_o :=gdist(G_1, G_2)
 4: ℓ_b := ℓ_o
 5: d_b := d_o
 6: t := t_0
 7: while t > t_f do
 8:    repeat
 9:       ℓ_c := ℓ_o
10:       r_1 :=irand(1, n)
11:       r_2 :=irand(1, n)
12:       k := ℓ_c^(r_1)
13:       ℓ_c^(r_1) := ℓ_c^(r_2)
14:       ℓ_c^(r_2) := k
15:    until ∄ i ∈ (1, ..., n) : κ_1(i) ≠ κ_2(ℓ_c(i))
16:    d_Δ :=gddelta(G_1, G_2, r_1, r_2)
17:    if urand(0, 1) ≤ exp(−d_Δ / t) then
18:       ℓ_o := ℓ_c
19:       d_o := d_o + d_Δ
20:       if d_o < d_b then
21:          ℓ_b := ℓ_o
22:          d_b := d_o
23:       end if
24:    end if
25:    t := γt
26: end while
27: return d_b, ℓ_b
```

the average complexity of the subroutine given here is still within the $\mathcal{O}(n^2)$ complexity of `gdist`. Likewise, the call to `gddelta` is $\mathcal{O}(n)$. Thus, `sdist` as written is still $\mathcal{O}(n^2)$ overall for fixed t_o, t_f, γ.

It should be noted that one obvious modification of the above algorithm is to evaluate the entire set of local moves (i.e., possible dyadic exchanges) at each iteration, choosing moves in the direction of steepest ascent.[5] Informal experimentation with this modified algorithm suggests that its convergence properties are excellent, but since the number of moves which must be evaluated at each step scales with $\mathcal{O}(n^2)$, this can prove prohibitively expensive for large graphs. (The total time complexity for this variant is $\mathcal{O}(n^3)$.) Other potentially useful modifications include changing the initial candidate labeling, and using a model for rejection probabilities other than the Boltzmann distribution of line 17. A thorough consideration of the impact of these changes on algorithm performance is an important topic for future research.

Where our analysis calls for the comparison of multiple graphs, it is generally useful to compute a matrix containing the structural distances between all graph pairs. The computation of such a distance matrix is a straightforward application of the `sdist` procedure, and for m graphs the complexity of such a computation is trivially $\mathcal{O}(m^2n^2)$.

3. Calculating Structural Covariances

The use of correlations and covariances to compare networks dates at least to the 1980s (Krackhardt, 1987a; Krackhardt, 1988).[6] This approach, however, has been limited in application to graph sets with unique common labelings, and attempts to generalize it beyond linear regression and bivariate correlation are have been limited. Butts and Carley (2001) have sought to extend this notion to the general (arbitrarily labeled) case, and to provide a clear basis for generalized linear subspace analysis of graph sets. Here, we provide some simple algorithms which implement these ideas.

The structural covariance builds straightforwardly on the graph covariance, itself merely the covariance of the adjacency matrices of two given graphs (i.e., $\text{cov}(\mathbf{vec}(\mathbf{A}_1), \mathbf{vec}(\mathbf{A}_2))$). Computation of the graph covariance is straightforward, and the `gcov` procedure shown in Algorithm 4 gives a typical "two-pass" implementation. The complexity of `gcov` is $\mathcal{O}(n^2)$. Where the graph correlation is desired, this can be obtained from the expression $\rho_{12} = \frac{\sigma_{12}}{\sqrt{\sigma_{11}\sigma_{22}}}$. As one would expect, σ_{ii} is the graph variance, and $\sigma_{ii} = n^2\delta_i(1-\delta_i)$ for dichotomous edge sets (where δ_i is the density). Note that the above expression is normalized for directed graphs, and includes loops; these should be omitted from the calculations in lines 5–6, 13–14, and their counts deducted from the normalizing constants in lines 11–12, 18 where edges are missing or undefined.

The classical graph covariance is defined only for graphs with a unique common labeling; as with the structural distance, treatment of the broader class of comparison problems requires the ability to consider the non-unique case. Butts and Carley (2001) have shown that a natural generalization of the graph covariance under non-unique common labelings is obtained by taking the maximum graph covariance over the set of all such labelings. This quantity is referred to as the *structural covariance*. Computation of the structural covariance poses the same challenges as does computation of the structural distance, and, in fact, similar algorithms may be employed for both tasks.

Here, we illustrate the use of a simulated annealing routine to estimate the structural covariance. As with the case of structural distance, this algorithm requires comparing the covariances induced be multiple labelings; substantial efficiency gains can be had by employing a routine like that shown in Algorithm 5 to compute change scores, rather than using multiple calls to `gcov`. In particular, let G_1 and G_2 be two graphs, and a, b two vertices to be exchanged under labeling ℓ. Then the quantity $(n^2-1)(\texttt{gcov}(G_1, \ell(G_2)) - \texttt{gcov}(G_1, G_2))$ can be obtained by means of `gcdelta` in $\mathcal{O}(n)$ time. Note that Algorithm 5 assumes that its input graphs have been demeaned prior to being passed to `gcdelta`—that is, the graph mean is assumed to have been subtracted from the value of each edge (including edges of value 0). For convenience, we will refer to a procedure which takes a single graph as input and returns the corresponding demeaned graph by `gdemean`. Since it must visit each dyad

Algorithm 4: Graph Covariance

procedure gcov(G_1, G_2)
$\mu_1 := 0$
$\mu_2 := 0$
$\sigma_{12} := 0$
for $i \in (1, \ldots, n)$ **do**
  for $j \in (1, \ldots, n)$ **do**
    $\mu_1 := \mu_1 + \mathbf{A}_1^{(ij)}$
    $\mu_2 := \mu_2 + \mathbf{A}_2^{(ij)}$
  end for
end for
$\mu_1 := \frac{\mu_1}{n^2}$
$\mu_2 := \frac{\mu_2}{n^2}$
for $i \in (1, \ldots, n)$ **do**
  for $j \in (1, \ldots, n)$ **do**
    $\sigma_{12} := \sigma_{12} + (\mathbf{A}_1^{(ij)} - \mu_1)(\mathbf{A}_2^{(ij)} - \mu_2)$
  end for
end for
$\sigma_{12} := \frac{\sigma_{12}}{n^2 - 1}$
return σ_{12}

Algorithm 5: Change in Graph Sums of Squares Under Single Dyad Permutation (Demeaned Inputs)

procedure gcdelta(G_1, G_2, a, b)
$\sigma := 0$
for $i \in (1, \ldots, n) \setminus \{a, b\}$ **do**
  $\sigma := \sigma + (A_1^{(ai)} - A_1^{(bi)})(A_2^{(bi)} - A_2^{(ai)}) + (A_1^{(ia)} - A_1^{(ib)})(A_2^{(ib)} - A_2^{(ia)})$
end for
$\sigma := \sigma + (A_1^{(aa)} - A_1^{(bb)})(A_2^{(bb)} - A_2^{(aa)}) + (A_1^{(ab)} - A_1^{(ba)})(A_2^{(ba)} - A_2^{(ab)})$
return σ

at least once, gdemean is clearly $\mathcal{O}(n^2)$; demeaning need only be performed once per input graph, however.

Given the above, structural covariance calculation may be accomplished through procedure scov (Algorithm 6), which takes as inputs two graphs (and their respective coloring functions) and returns the structural covariance along with the associated labeling function. The internal parameters of scov are identical to those of sdist, and the same considerations apply when tuning the algorithm; note that one difference between the two procedures is that the former acts to *maximize* the criterion, while the latter *minimizes*

it. Thus, the sense of lines 19 and 22 has been reversed here. Since `scov` must carry out an initial covariance calculation (and demeaning operation), it is at least $\mathcal{O}(n^2)$. The internal annealing loop in lines 9–28 is of the same complexity as the corresponding loop in Algorithm 3, and the same comments with respect to the label drawing subroutine (lines 10–17) apply. As with `sdist`, an obvious modification of Algorithm 6 considers all permissible dyadic exchanges at each iteration, taking the most favorable alternative as the candidate draw. This is an $\mathcal{O}(n^3)$ algorithm, but may be efficacious in practice.

As with the graph covariance, the structural covariance of a graph with itself is the graph variance. Since this last is a labeling-invariant property, it follows that only one labeling

Algorithm 6: Structural Covariance

1: **procedure** `scov`$(G_1, G_2, \kappa_1, \kappa_2)$
2: $\ell_o := (1, \ldots, n)$
3: $\sigma_o := (n^2 - 1)$`gcov`(G_1, G_2)
4: $\ell_b := \ell_o$
5: $\sigma_b := \sigma_o$
6: $H_1 :=$`gdemean`(G_1)
7: $H_2 :=$`gdemean`(G_2)
8: $t := t_0$
9: **while** $t > t_f$ **do**
10: **repeat**
11: $\ell_c := \ell_o$
12: $r_1 :=$`irand`$(1, n)$
13: $r_2 :=$`irand`$(1, n)$
14: $k := \ell_c^{(r_1)}$
15: $\ell_c^{(r_1)} := \ell_c^{(r_2)}$
16: $\ell_c^{(r_2)} := k$
17: **until** $\nexists\, i \in (1, \ldots, n) : \kappa_1(i) \neq \kappa_2(\ell_c(i))$
18: $\sigma_\Delta :=$`gcdelta`(H_1, H_2, r_1, r_2)
19: **if** `urand`$(0, 1) \leq \exp(\frac{\sigma_\Delta}{t})$ **then**
20: $\ell_o := \ell_c$
21: $\sigma_o := \sigma_o + \sigma_\Delta$
22: **if** $\sigma_o > \sigma_b$ **then**
23: $\ell_b := \ell_o$
24: $\sigma_b := \sigma_o$
25: **end if**
26: **end if**
27: $t := \gamma t$
28: **end while**
29: $\sigma_b := \frac{\sigma_b}{n^2-1}$
30: **return** σ_b, ℓ_b

search is needed to compute the structural correlation between two graphs: one finds the structural covariance by heuristic search and then divides by the square root of the product of the respective graph variances. The structural correlation is itself an obvious generalization of the graph correlation (Krackhardt, 1987b), and is useful when one seeks a standardized measure of association between graphs.

Like the structural distance, structural covariances are most commonly employed as components of a matrix of such quantities. These structural covariance matrices are directly analogous to conventional covariance matrices, and have been shown to possess similar properties (Butts and Carley, 2001). In particular, structural covariance matrices are positive semi-definite, and hence have easily interpretable eigenstructures. Computation of structural covariance matrices is straightforward, although some efficiency gains can be obtained by pre-demeaning all graphs and skipping the demeaning step in Algorithm 6. For a set of m graphs, a complete covariance matrix requires $\binom{m}{2}$ calls to `scov`, and the resulting algorithm is hence $\mathcal{O}(m^2n^2)$. Observe that the diagonal entries of the structural covariance matrix are the respective graph variances, and that a structural covariance matrix can be transformed into a structural correlation matrix via the appropriate normalization of each entry.

4. Conclusion

In this paper we have provided simple algorithms which implement the various graph comparison measures presented in Butts and Carley (2001). Specifically, we have shown heuristic optimization procedures which allow the approximate calculation of structural distances, covariances, and correlations in quadratic time. While it is quite likely that the algorithms shown here can be improved upon, our purpose has been to provide a starting point for those who wish to implement these methods for their own use. It is hoped that by detailing one such set of implementations, others will be motivated to experiment with and improve the underlying framework.

Acknowledgment

This work was supported in part by: the Office of Naval Research (ONR), United States Navy grant No. 1681-1-1001944; the Army Research Labs under the grant "Personnel Turnover as Team Performance"; the National Science Foundation (NSF) under a Graduate Research Fellowship, grant No. ITR/IM IIS-0081219, ITR IIS-0331707 and the CASOS IGERT grant; and both the center for the Computational Analysis of Social and Organizational Systems (CASOS) and the Institute for Complex Engineered Systems (ICES) at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, express or implied, of the ONR, the NSF, or the US Government. The authors would also like to thank two anonymous reviewers for their helpful suggestions.

Notes

1. Simple implementations of these methods in R and C are obtainable from the authors upon request.
2. For purposes of the present work, we take all graphs to be of the same order. Numerous strategies are available for coping with graphs of mixed order; see Butts and Carley (2001) for a discussion.
3. We do this because, in practice, ℓ is likely to be stored as a vector, and hence this notation is in keeping with our other usage.
4. The set of common labelings is referred to in Butts and Carley (2001) as the "accessible permutation group."
5. Care must be taken, however, not to automatically select candidate labelings of equal performance when better labelings are not available, since this can easily lead to looping. Such a difficulty does not surface when a single candidate is drawn at each iteration, since the probability of choosing the preceding labeling falls rapidly in n.
6. For a much older example of the use of matrix correlations in cluster analysis, see Farris (1969). See also Hubert (1987) for a review of related methods.

References

Abbott, A. and A. Tsay (2000), "Sequence Analysis and Optimal Matching Methods in Sociology: Review and Prospect," *Sociological Methods and Research*, 29:3–33.

Banks, D. and K.M. Carley (1994), "Metric Inference for Social Networks," *Journal of Classification*, 11(1), 121–149.

Borg, I. and J. Lingoes (1987), *Multidimensional Similarity Structure Analysis*., Springer-Verlag, New York.

Butts, C.T. (1998), Cluster analysis of unlabeled structures. ICES Research Report 88-04-98, Institute for Complex Engineered Systems, Carnegie Mellon University.

Butts, C.T. and K.M. Carley (1998), "Canonical Labeling to Facilitate Graph Comparison," ICES Research Report 88-06-98, Institute for Complex Engineered Systems, Carnegie Mellon University.

Butts, C.T. and K.M. Carley (2001), "Multivariate Methods for Interstructural Analysis," CASOS working paper, Center for the Computational Analysis of Social and Organization Systems, Carnegie Mellon University.

Farris, J.S. (1969), "On the Cophenetic Correlation Coefficient," *Systematic Zoology*, 18, 279–285.

Gentle, J.E. (1998), *Random Number Generation and Monte Carlo Methods*, Springer, New York.

Hamming, H. (1950), "Error Detecting and Error Correcting Codes," *Bell System Technical Journal*, 29, 147–160.

Hubert, L.J. (1987), *Assignment Methods in Combinatorial Data Analysis*, Marcel Dekker, New York.

Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi (1983), "Optimization by Simulated Annealing," *Science*, 220, 671–680.

Krackhardt, D. (1987a), "Cognitive Social Structures," *Social Networks*, 9, 109–134.

Krackhardt, D. (1987b), "QAP Partialling as a Test of Spuriousness," *Social Networks*, 9, 171–186.

Krackhardt, D. (1988), "Predicting With Networks: Nonparametric Multiple Regression Analyses of Dyadic Data," *Social Networks*, 10, 359–382.

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller (1953), "Equation of State Calculations by Fast Computing Machine," *Journal of Chemical Physics*, 21, 1087–1091.

Otten, R.H.J.M. and L.P.P.P. van Ginneken (1989), *The Annealing Algorithm*. Kluwer, Boston.

Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery (1992), *Numerical Recipes: The Art of Scientific Computing*., Cambridge University Press, Cambridge, second edition.

H.C. Romesburg (1984), *Cluster Analysis for Researchers*., Lifetime Learning Publications, Belmont, California.

W.S. Torgerson (1952), "Multidimensional Scaling: I, Theory and Method," *Psychometrika*, 17, 401–419.

Carter T. Butts is Assistant Professor at the University of California-Irvine in the Department of Sociology, and is a member of the Institute for Mathematical Behavioral Sciences and the California Institute for Telecommunications and Information Technology. His current research focuses on communication during disasters, Bayesian inference for network data, network comparison, and the structure of spatially embedded interpersonal networks.

Kathleen M. Carley is Professor at Carnegie Mellon University, with appointments in the Institute for Software Research International, the H.J. Heinz III School of Public Policy and Management, and the Department of Engineering and Public Policy. Her research centers around areas of social, organizational, knowledge and information networks, organizational design, change, adaptivity and and performance, computational organization theory, crisis management, social theory, impacts on information diffusion of changes in social policy and changes in communication technology, and mapping experts' and executives' knowledge networks using textual analysis techniques.