Vertex Collocation Profiles: Subgraph Counting for Link Analysis and Prediction

Ryan N. Lichtenwalter
University of Notre Dame
Notre Dame, Indiana, USA
rlichten@nd.edu

Nitesh V. Chawla
University of Notre Dame
Notre Dame, Indiana, USA
nchawla@nd.edu

ABSTRACT

We introduce the concept of a vertex collocation profile (VCP) for the purpose of topological link analysis and prediction. VCPs provide nearly complete information about the surrounding local structure of embedded vertex pairs. The VCP approach offers a new tool for domain experts to understand the underlying growth mechanisms in their networks and to analyze link formation mechanisms in the appropriate sociological, biological, physical, or other context. The same resolution that gives VCP its analytical power also enables it to perform well when used in supervised models to discriminate potential new links. We first develop the theory, mathematics, and algorithms underlying VCPs. Then we demonstrate VCP methods performing link prediction competitively with unsupervised and supervised methods across several different network families. We conclude with timing results that introduce the comparative performance of several existing algorithms and the practicability of VCP computations on large networks.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

Keywords

graph theory, link analysis, link prediction, network analysis

1. INTRODUCTION

Link prediction is the task of inferring links in a graph G_{t+1} based on the observation of a graph G_t. It may be that $t+1$ follows t in time, or it may be that $t+1$ represents some other evolution or manipulation of the graph such as including additional links from experiments that are difficult or expensive to conduct. Link prediction stated in this manner is a binary classification problem in which links that form construct the positive class and links that do not form construct the negative class. Link analysis, more loosely defined, is the problem of identifying evolutionary processes or growth mechanisms in a network that are responsible for the formation of new relationships between nodes.

We formally define a new technique for performing both link prediction and link analysis based on a restrictive representation of the local topological embedding of the source and target vertices. This idea is a generalization and extension of the triangle counting approach for multi-relational prediction in [6]. It also draws on concepts from literature on graphlets as introduced in [19] and to a lesser degree from motif analysis as discussed in [17].

Many existing link prediction models compress a selection of simple information in theoretically or empirically guided ways. By contrast the VCP approach preserves as much topological information as possible about the embedding of the source and target vertices. It also extends naturally to multi-relational networks and can thereby encode a variety of additional information such as edge directionality. It can encode continuous quantities such as edge weights by binning into relational categories, such as high activity and low activity. Information about the nature of relationships is maintained as structures are identified within the network. We proceed with a formal exploration of VCP, discuss its relationship to isomorphism classes, provide algorithms that formally describe VCP computations, and demonstrate the potential of VCP in link prediction and analysis as well as feasibility in terms of computational time. Fast forms of the algorithms listed within this paper are all implemented in C++ and integrated into the LPmade [14] link prediction software and are thus freely available on MLOSS.

2. VERTEX COLLOCATION PROFILES

Formally, a *vertex collocation profile* (VCP), written as $\text{VCP}^{n,r}_{i,j}$, is a vector describing the relationship between two vertices, v_i and v_j, in terms of their common membership in all possible subgraphs of n vertices over r relations. A VCP *element*, $\text{VCP}^{n,r}_{i,j}(x)$ is defined as a distinct embedding of v_i and v_j within a particular isomorphism class of n vertices and is represented by a uniquely addressable cell in the VCP vector. Figure 1 illustrates the first 16 elements of $\text{VCP}^{3,2}_{s,t}$, where the two relations correspond to edge directionality.

In general, we can encode the connectivity in any multi-relational network of r relations with 2^r different types of connections. We use 2^r instead of 2^{r-1} because structural holes are often as important as links [5], and we consider the lack of relation as itself a type of connection. Undirected single-relational networks exhibit two types of connections: existent and nonexistent. Directed single-relational networks are similar to undirected bi-relational networks and have four types of connections: nonexistent, outward, inward, and bidirectional.

The cardinality of $\text{VCP}^{n,r}$ depends upon the number of vertices n and the number of types of relationship r in the set of relations $\mathcal{R}$. The space grows exponentially in the number

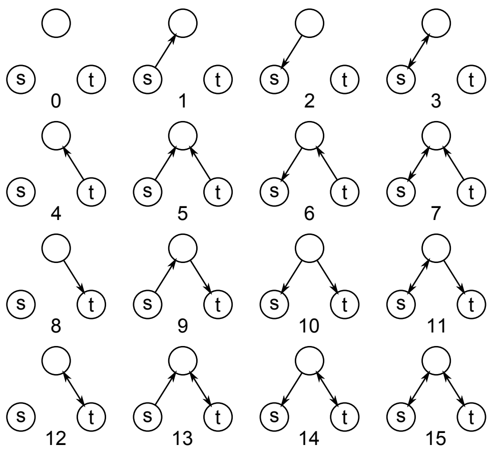


Figure 1: Elements of $\mathbf{VCP}^{3,2}_{s,t}$. 16 through 31 are identical except with the presence of $e_{t,s}$.

$n \backslash r$	1	2	3	4
3	4	32	256	2048
4	32	2048	131072	8.4×10^6
5	512	524288	5.4×10^8	5.5×10^{11}
6	16384	5.4×10^8	1.8×10^{13}	5.8×10^{18}

Table 1: Number of enumerated subgraphs composing VCP for values of n and r.

of vertices with the base as the cardinality of the power set of relations. The formula for the number of subgraphs is written in intuitive form in Equation 1. The multiplier accounts for the number of possible collocation structures disregarding any links between the source and the target. The multiplicand is the number of different ways two vertices with the same embedding can appear based on the different link possibilities between them.

$$(2^r)^{\left(\frac{n(n-1)}{2}-1\right)} \times 2^{r-1} \tag{1}$$

We can manipulate these to achieve the simpler formula below.

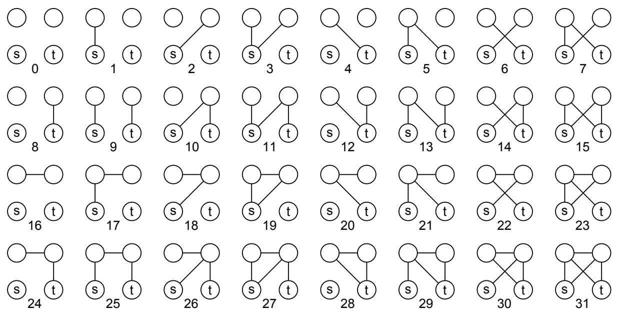
$$2^{\frac{n(n-1)r}{2}-1} \tag{2}$$

Table 1 illustrates the number of subgraphs respecting vertex identity that compose the VCP given different values of n and r.

The number of subgraphs grows at such a rate as to make the sheer size of output unmanageable for large values of n and r. The rate of growth of VCPs is much slower due to superlinear increases in the isomorphisms with increasing n, but nonetheless VCP cardinality also grows quickly. Fortunately, the most important information is typically located close to the source and target vertices, and many networks have few types of relationships. When the number of relationship types is high, relationships can be compressed or discarded in various ways albeit with a loss of information.

2.1 Isomorphisms

Isomorphic subgraphs are closely related to VCP elements. In Figure 2, it is impossible to distinguish subgraph 1 from



0: {0}	5: {6,9}	10: {16}	15: {22,25}
1: {1,2}	6: {7,11}	11: {17,18}	16: {23,27}
2: {3}	7: {12}	12: {19}	17: {28}
3: {4,8}	8: {13,14}	13: {20,24}	18: {29,30}
4: {5,10}	9: {15}	14: {21,26}	19: {31}

Figure 2: Subgraphs that form $\mathbf{VCP}^{4,1}_{s,t}$ and the mapping of isomorphic subgraphs to VCP elements.

subgraph 2, and the prevalence of the frequency of each during counting will depend upon implementation details of the counting algorithm that determine the order in which the vertices are selected. These subgraphs map to the same VCP element, $\text{VCP}^{4,1}_{s,t}(1)$, and the count of that element is the sum of the counts of the isomorphic subgraphs.

Isomorphisms that require a mapping between v_s and v_t, for instance subgraph 1 and subgraph 8 in Figure 2, do not share the same VCP element even though they reside within the same isomorphism class. VCP elements ignore isomorphisms that require mapping v_s to v_t because VCP describes the local embedding of these two explicitly identified vertices. In undirected graphs, elements such as $\text{VCP}^{4,1}_{s,t}(1)$ and $\text{VCP}^{4,1}_{s,t}(3)$ together supply information regarding symmetry or asymmetry in the density of the embedding of v_s and v_t. The distinction in directed networks is more obvious and relates to the potential significance of the difference in the local topologies of the source of a new link and its target. Figure 2 shows all the subgraphs pertinent to $\text{VCP}^{4,1}$ and their corresponding mappings to elements.

Counting the number of elements in $\text{VCP}^{n,r}$ is related to the complex problem of counting the number of isomorphism classes in graphs of n vertices. In $\text{VCP}^{3,r}$, each enumerated subgraph maps uniquely to a VCP element. v_s and v_t are fixed, and there is only one permutation of the remaining vertex. In $\text{VCP}^{4,r}$, there are two mappable vertices and the number of $\text{VCP}^{4,r}$ elements is described as:

$$2^{6r-2} + 2^{4r-2} \tag{3}$$

The derivation of a general formula for $|\text{VCP}^{n,r}|$ for all n and r is extremely combinatorially involved and its discussion is beyond the scope of this paper. We have instead provided software that computes VCP cardinalities and subgraph-to-element mappings for $3 < n < 8$ and $1 < r < 8$. For practical purposes, Table 2 shows the cardinality of all VCPs with fewer than a million elements.

2.2 Addressing

We define a VCP addressing scheme similar to the isomorphism certificate addressing scheme in [12]. The subgraphs from which the elements are derived are indexed by assign-

Table 2: Number of elements in $\mathbf{VCP}_{i,j}^{n,r}$.

n \ r	1	2	3	4	5
3	4	32	256	2048	16384
4	20	1088	66560	-	-
5	120	91520	-	-	-
6	996	-	-	-	-
7	12208	-	-	-	-

ing powers of 2^r to edges in the adjacency matrix in increasing lexicographical order starting with $e_{1,3}$ and ending with $e_{n-1,n}$. v_s and v_t are defined as v_1 and v_2 respectively, and $e_{1,2}$ is the edge of highest value. The value of each edge is multiplied by the index of the lexicographically ordered power set, $\mathcal{P}(\mathcal{R})$, corresponding to the ordered set of $\mathcal{R}$ relations on the edge. Figures 1 and 2 demonstrate the indexing scheme for two different values of n and r. For any selection of vertices $v_s, v_t, v_3, ..., v_n$, this addressing scheme will map the resulting multi-relational subgraph to an index that exists within a set of indices of isomorphic structures.

We define the unique *address* of a VCP element as the subgraph representative with the lowest index within the corresponding isomorphism class. This addressing scheme provides a unique address for all elements in all VCPs. The addresses for elements in VCP^{4,1} are provided in Figure 2. Because manual identification of isomorphism classes is error-prone and difficult especially as the number of subgraphs increases, we have provided a program that outputs the mapping from all subgraph indices to their corresponding element addresses for all VCPs.

2.3 Directionality

Directed networks with r relations can be treated similarly to undirected networks with $2r$ relations with one major caveat. The subgraph-to-element mapping differs with directed networks. Taking $e_{i,j}^x$ momentarily as notation for an edge of relation x and $2x$ as relation x in the opposite direction, $e_{i,j}^x \equiv e_{j,i}^x$ in undirected multi-relational networks, but $e_{i,j}^x \equiv e_{j,i}^{2x}$ in directed networks. In the context of larger subgraphs, this causes more isomorphic equivalences and decreases the cardinality of the VCP by comparison to its undirected pseudo-equivalent, a fact demonstrated in Figure 3. For instance, VCP^{4,2} contains 1088 elements whereas the directed variant of VCP^{4,1} contains only 1056 elements.

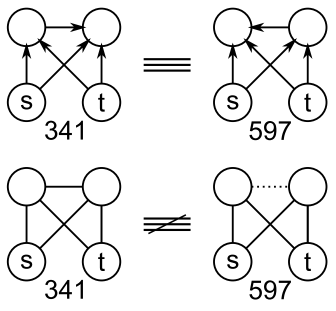


Figure 3: Subgraphs from undirected VCP^{4,2} and directed VCP^{4,1}. The directed subgraphs both map to directed VCP^{4,1}(221), but the undirected subgraphs map to two different VCP^{4,2} elements.

Algorithm 1 VCP^{3,r}

```
Input: network $G = (V, \boldsymbol{E})$,
    relations $\mathcal{R}$,
    $i : v_i \in V$,
    $j : v_j \in V$
Output: $VCP_{i,j}^{3,|\mathcal{R}|}$
1: $\sigma_{i,j} \leftarrow \Phi(\mathcal{P}(\mathcal{R}), e_{i,j})$
2: for $k : e_{i,k} \in E_i \wedge e_{j,k} \in E_j$ do
3:     $\sigma_{i,k} \leftarrow \Phi(\mathcal{P}(\mathcal{R}), e_{i,k})$
4:     $\sigma_{j,k} \leftarrow \Phi(\mathcal{P}(\mathcal{R}), e_{j,k})$
5:     $\lambda \leftarrow 2^{2|\mathcal{R}|}\sigma_{i,j} + 2^{|\mathcal{R}|}\sigma_{j,k} + \sigma_{i,k}$
6:     $VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] \leftarrow VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] + 1$
7: end for
8: for $k : e_{i,k} \in E_i \wedge e_{j,k} \notin E_j$ do
9:     $\sigma_{i,k} \leftarrow \Phi(\mathcal{P}(\mathcal{R}), e_{i,k})$
10:    $\lambda \leftarrow 2^{2|\mathcal{R}|}\sigma_{i,j} + 2^{|\mathcal{R}|}\sigma_{j,k} + \sigma_{i,k}$
11:    $VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] \leftarrow VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] + 1$
12: end for
13: for $k : e_{i,k} \notin E_i \wedge e_{j,k} \in E_j$ do
14:    $\sigma_{j,k} \leftarrow \Phi(\mathcal{P}(\mathcal{R}), e_{j,k})$
15:    $\lambda \leftarrow 2^{2|\mathcal{R}|}\sigma_{i,j} + 2^{|\mathcal{R}|}\sigma_{j,k} + \sigma_{i,k}$
16:    $VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] \leftarrow VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] + 1$
17: end for
18: $\lambda \leftarrow 2^{2|\mathcal{R}|}\sigma_{i,j}$
19: for $k : e_{i,k} \notin E_i \wedge e_{j,k} \notin E_j$ do
20:    $VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] \leftarrow VCP_{i,j}^{3,|\mathcal{R}|}[\lambda] + 1$
21: end for
22: return $VCP_{i,j}^{3,|\mathcal{R}|}$
```

Therefore, the algorithms in Section 3 and the procedures described by all provided code work with only minor adjustments, which are essentially related to the subgraph-to-element mapping. We include software to construct the mapping for VCP^{4,1} where there are actually two relations corresponding to directionality.

3. ALGORITHMS

Algorithms 1 and 2 serve as reference algorithms and not as optimized or even asymptotically optimal approaches for VCP element counting. In fact, implementations of the naïve VCP^{4,1} algorithm fail to return within a reasonable time for networks with even thousands of nodes. Fortunately, it is possible to design much faster approaches, and we implemented these approaches and provide them as a set of C++ files. We also present a more innovative algorithm, Algorithm 3 for counting VCP^{4,1} that corresponds to the approach in the code for that VCP.

3.1 3-Vertex VCP

Algorithm 1 demonstrates how to calculate VCP^{3,r} for the set of r relations in $\mathcal{R}$. $\Phi(\mathcal{P}(\mathcal{R}), e_{x,y})$ refers to a procedure to determine the index of the multi-relational edge $e_{x,y}$ in $\mathcal{P}(\mathcal{R})$, the lexicographically ordered power set of relations. This procedure can derive power set indices efficiently by setting individual bits in the index according to the presence of the relation corresponding to that bit and indexing the bits by the natural order of the relations themselves.

This naïve algorithm first determines the contribution of any edge types between v_i and v_j, the fixed source and target vertices for the link. Then it counts subgraphs with a third vertex connected to both v_i and v_j, subgraphs only connected to v_i, subgraphs only connected to v_j, and both. σ represents the identity of an edge within $\mathcal{P}(\mathcal{R})$ and λ represents the index of the subgraph. Because no isomorphisms exist with only one mappable vertex, the algorithm directly increments the VCP elements corresponding to the underlying subgraph index as contrasted to Algorithm 2 wherein the subgraph index must be mapped to an element address.

A naïve implementation iterates through each vertex in the network and determines the corresponding subgraph index by summing the edge contributions. For one free vertex, this approach has complexity $O\left(|V|\log\left(\frac{|E|}{|V|}\right)\right)$ per edge output assuming $O\left(\log\left(\frac{|E|}{|V|}\right)\right)$ neighbor search time for the average case. This complexity is probably feasible for small networks, but may require an unacceptably long time for large networks. It is simple to improve upon this approach by considering only the vertices that are neighbors, denoted by $\Gamma(v_x)$ of v_i, v_j, both, or neither and performing set operations. $\mathrm{VCP}^{3,1}_{i,j}(0)$ is populated by subtracting $|\Gamma(v_i)\cup\Gamma(v_j)|$ from $|V|-2$. $\mathrm{VCP}^{3,1}_{i,j}(1)$ and $\mathrm{VCP}^{3,1}_{i,j}(2)$ are populated by computing set differences $|\Gamma(v_i)-\Gamma(v_j)|$ and $|\Gamma(v_j)-\Gamma(v_i)|$ respectively. $\mathrm{VCP}_{i,j}$ is computed as the intersection $|\Gamma(v_i)\cap\Gamma(v_j)|$. These operations can be performed quickly especially in graphs in which adjacencies are represented as ordered lists of neighbors. This implementation has average-case complexity $O\left(\frac{|E|}{|V|}\right)$ per edge output.

3.2 4-Vertex VCP

Algorithm 2 iterates through every pair of free vertices, yielding a complexity of $O\left(|V|^2\log\left(\frac{|E|}{|V|}\right)\right)$ from $\binom{|V|-2}{2}$ pairs of free vertices. This requires trillions of operations even for small networks. It is possible to reduce this time greatly by restricting consideration to known neighbors as described in the discussion of Algorithm 1, but naïve implementations of this optimization involve many expensive operations in hashes or balanced search trees.

Algorithm 3 instead uses a minimal number of set operations implemented as merge operations on ordered lists. Figure 4 provides an illustration of the sets mentioned in the following explanation. First, the number of connected pairs and unconnected pairs is computed once for the entire network, and these values are represented as χ_G and ϵ_G respectively. We must also track the connected pairs and unconnected pairs internal to the vertices in our consideration for the prediction output to differentiate $\mathrm{VCP}^{4,1}(0)$ from $\mathrm{VCP}^{4,1}(10)$. We start by constructing a set of potential "third position" vertices, Γ_3, as $\Gamma(v_i)\cap\Gamma(v_j)$. For each member of Γ_3, we construct two disjoint sets of "fourth position" vertices, Γ_4 containing vertices reachable by our current member of Γ_3 but not contained within Γ_3, and Γ_{4a} constructed from Γ_3 excluding the current member of Γ_3. In Γ_4, we count new connections and gaps in the configuration, and we increment the counter for the corresponding subgraph. For Γ_{4a}, we do not count connections and gaps since configurations using those set members are counted when they serve as members of Γ_3, or "third position" vertices. Likewise, we only count subgraphs with two members from Γ_3 when the member from Γ_{4a} compares lower. This avoids multi-counting. After considering the configurations from all members of Γ_4 and Γ_{4a}, we account for

Algorithm 2 $\mathrm{VCP}^{4,1}$ (Simple)

```
Input: network $G = (V, E)$,
    $i : v_i \in V$,
    $j : v_j \in V$,
    subgraph-element mapping $M$
Output: $VCP^{4,1}_{i,j}$

 1: for $k : v_k \neq v_i \wedge v_k \neq v_j$ do
 2:   for $l : v_l \neq v_i \wedge v_l \neq v_j \wedge l > k$ do
 3:     $\lambda \leftarrow 0$
 4:     if $e_{i,k} \in E$ then
 5:       $\lambda \leftarrow \lambda + 1$
 6:     end if
 7:     if $e_{i,l} \in E$ then
 8:       $\lambda \leftarrow \lambda + 2$
 9:     end if
10:     if $e_{j,k} \in E$ then
11:       $\lambda \leftarrow \lambda + 4$
12:     end if
13:     if $e_{j,l} \in E$ then
14:       $\lambda \leftarrow \lambda + 8$
15:     end if
16:     if $e_{k,l} \in E$ then
17:       $\lambda \leftarrow \lambda + 16$
18:     end if
19:     $VCP^{4,1}_{i,j}(M(\lambda)) \leftarrow VCP^{4,1}_{i,j}(M(\lambda)) + 1$
20:   end for
21: end for
22: return $VCP^{4,1}_{i,j}$
```

structures with isolates by contributing $|V|-2-|\Gamma_3|-|\Gamma_4|$ to $VCP^{4,1}_{i,j}(M(\lambda_1))$. We also account for multi-counting of $\mathrm{VCP}^{4,1}(0)$ due to duplicate consideration of gaps by subtracting the same quantity from $VCP^{4,1}_{i,j}(M(0))$. Finally, we compute $\mathrm{VCP}^{4,1}(0)$ and $\mathrm{VCP}^{4,1}(10)$ using our computations of vertices and gaps in the vertices we have encountered in the single Γ_3 and multiple Γ_4 sets and subtract their contributions from the contributions from the entire network. It is possible to perform the entire procedure using a few relatively inexpensive merge operations in ordered vectors or lists and entirely avoiding hashes or balanced trees. This exposition mostly describes the procedure to quickly compute $\mathrm{VCP}^{4,1}$ albeit omitting minor implementation details. We refer more interested readers to the code itself.

3.3 Extension to Complex Networks

It is trivial to extend VCP algorithms to networks more complex than those on which we obtained our results. This includes any form of edge feature such as directionality, weight, temporality, different relation types, or any information describing edges or vertex pairs that either exists categorically or can be quantized. One amenable network representation associates an ordered set of bits with each edge. Each bit corresponds to the presence of a particular relation or some Boolean descriptor for a pair of vertices. The determination of the existence of an edge for single-relational data instead becomes an evaluation of the edge as the binary-coded integral value of the ordered set of bits. This is one conceivable implementation for $\Phi(\mathcal{P}(\mathcal{R}), e_{x,y})$ in Algorithm 1, which replaces the constant values for all λ updates in Algorithms 2 and 3. For most values of r, this

Algorithm 3 $\text{VCP}^{4,1}$ (Fast)

```
Input: network $G = (V, E)$,
    $i : v_i \in V$,
    $j : v_j \in V$,
    subgraph-element mapping $M$
    count of connected pairs $\chi_G$,
    count of unconnected pairs $\epsilon_G$
Output: $VCP^{4,1}_{i,j}$

1: $\upsilon \leftarrow 2$
2: $\chi \leftarrow 0$
3: $\epsilon \leftarrow 0$
4: $\Gamma_3 \leftarrow \Gamma(v_i) \cap \Gamma(v_j)$
5: for $k : v_k \in \Gamma_3$ do
6:     $\lambda_1 \leftarrow 0$
7:     if $e_{i,k} \in E \vee e_{k,i} \in E$ then
8:         $\lambda_1 \leftarrow \lambda_1 + 1$
9:         $\chi \leftarrow \chi + 1$
10:    else
11:        $\epsilon \leftarrow \epsilon + 1$
12:    end if
13:    $\Gamma_4 \leftarrow \Gamma(v_k) - \Gamma_3$
14:    $\upsilon \leftarrow \upsilon + |\Gamma_4|$
15:    for $l : v_l \in \Gamma_4$ do
16:        $\lambda_2 \leftarrow \lambda_1$
17:        if $e_{i,l} \in E \vee e_{l,i} \in E$ then
18:            $\lambda_2 \leftarrow \lambda_2 + 2$
19:            $\chi \leftarrow \chi + 1$
20:        else
21:            $\epsilon \leftarrow \epsilon + 1$
22:        end if
23:        if $e_{j,l} \in E \vee e_{l,j} \in E$ then
24:            $\lambda_2 \leftarrow \lambda_2 + 8$
25:            $\chi \leftarrow \chi + 1$
26:        else
27:            $\epsilon \leftarrow \epsilon + 1$
28:        end if
29:        if $e_{k,l} \in E \vee e_{l,k} \in E$ then
30:            $\lambda_2 \leftarrow \lambda_2 + 16$
31:            $\chi \leftarrow \chi + 1$
32:        else
33:            $\epsilon \leftarrow \epsilon + 1$
34:        end if
35:        $VCP^{4,1}_{i,j}(M(\lambda_2)) \leftarrow VCP^{4,1}_{i,j}(M(\lambda_2)) + 1$
36:    end for
37:    for $l : v_l \in \Gamma_3 \wedge l > k$ do
38:        $\lambda_2 \leftarrow \lambda_1$
39:        if $e_{i,l} \in E \vee e_{l,i} \in E$ then
40:            $\lambda_2 \leftarrow \lambda_2 + 2$
41:        end if
42:        if $e_{j,l} \in E \vee e_{l,j} \in E$ then
43:            $\lambda_2 \leftarrow \lambda_2 + 8$
44:        end if
45:        if $e_{k,l} \in E \vee e_{l,k} \in E$ then
46:            $\lambda_2 \leftarrow \lambda_2 + 16$
47:            $\chi \leftarrow \chi + 1$
48:        else
49:            $\epsilon \leftarrow \epsilon + 1$
50:        end if
51:        $VCP^{4,1}_{i,j}(M(\lambda_2)) \leftarrow VCP^{4,1}_{i,j}(M(\lambda_2)) + 1$
52:    end for
53:    $\zeta \leftarrow |\Gamma_3| + |\Gamma_4|$
54:    $VCP^{4,1}_{i,j}(M(\lambda_1)) \leftarrow VCP^{4,1}_{i,j}(M(\lambda_1)) + |V| - 2 - \zeta$
55:    $VCP^{4,1}_{i,j}(M(0)) \leftarrow VCP^{4,1}_{i,j}(M(0)) + |V| - 2 - \zeta$
56: end for
57: $\rho \leftarrow e_{i,j} \in E \vee e_{j,i} \in E$
58: $VCP^{4,1}_{i,j}(M(16)) \leftarrow VCP^{4,1}_{i,j}(M(16)) + \chi_G - (\chi + \rho)$
59: $VCP^{4,1}_{i,j}(M(0)) \leftarrow VCP^{4,1}_{i,j}(M(0)) + \epsilon_G - (\epsilon + \neg(\rho)) - (2|V| - \upsilon)$
60: return $VCP^{4,1}_{i,j}$
```

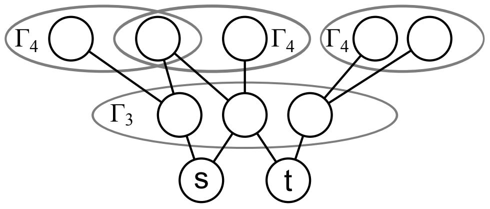


Figure 4: A depiction of the sets considered within Algorithm 3.

can be implemented as a constant-time operation equivalent to retrieving the value of a variable, so the asymptotic cost of populating the VCP vector is unaffected. Excepting the additional costs of writing output and of allocating and deallocating the storage necessary to hold the additional multi-relational structural elements, which is inexpensively proportional to 2^r, the computational complexity of the multi-relational extension is no greater than for single-relational networks.

4. RESULTS

First, we illustrate how VCP can serve as a powerful link analysis and modeling tool. Then we perform a standard comparison of the link prediction efficacy of VCP and a selection of other methods. Timing results are rarely provided in link prediction work despite vast differences in the running time and feasibility of methods. For this reason and because we believe many might suspect that a completely theoretically aligned implementation of VCP is computationally unachievable, we also provide comparative timing results.

4.1 Data

We present results for several different data sets to demonstrate the performance of the techniques under comparison for different families of networks. Though all of these data sets contain information with which to generate edge weights, we are interested in providing purely structural comparison here, so all quantitative results are presented based on networks constructed without edge weights.

`calls` is a stream of 262 million cellular phone calls from a major cellular phone service provider. We construct directed networks from the calls by creating a node v_i for each caller and a directed link $e_{i,j}$ from v_i to v_j if and only if v_i calls v_j. `sms` is a stream of 84 million text messages from the same source as `calls` and constructed in the same manner. These two data sets are not publicly available.

`condmat-collab` is a stream of 19,464 multi-agent events representing condensed matter physics collaborations from 1995 to 2000. We construct undirected networks from the collaborations by creating a node for each author in the event and an undirected link connecting each pair of authors. For all experiments involving `condmat`, we use the years 1995 to 1999 for constructing training data and the year 2000 for testing.

`dblp-cite` is a citation network based on the DBLP computer science bibliography. Each researcher is a node v_i and directed networks are formed by viewing a citation by researcher v_i of work by researcher v_j as a directed link $e_{i,j}$. The `dblp-collab` network uses the same raw data, but links are based on co-authorship collaborations. An undirected

Table 3: Some basic properties of the data sets. These figures are reported for networks constructed using all available longitudinal data. $\overline{C}$ represents average clustering coefficient and r_a represents assortativity coefficient.

Name	Directed	Vertices	Edges	$\overline{C}$	r_a
calls	✓	7,786,471	33,292,508	0.127	0.212
condmat-collab		17,216	110,544	0.642	0.177
dblp-cite	✓	15,963	344,373	0.128	-0.046
dblp-collab		367,725	2,088,710	0.617	0.254
disease-g		399	15,634	0.665	-0.310
disease-p		437	81,158	0.818	-0.406
hepth-cite	✓	8,249	335,028	0.352	0.097
hepth-collab		8,381	40,736	0.466	0.237
huddle		4,243	997,008	0.591	-0.211
patents-collab		1,162,227	5,448,168	0.531	0.141
sms	✓	5,016,746	11,598,843	0.048	0.042

link exists between v_i and v_j if both are an author on the same paper.

`disease-g` is a network in which nodes represent diseases and the links between diseases represent the co-occurrence of particular genotypic characteristics. Links are undirected. This network is not longitudinal, but finding unobserved links is an important task, so we have no choice but to estimate performance by randomly removing links to construct test sets. `disease-p` is from the same source as `disease-g`. The difference is that the links in `disease-p` represent the co-occurrence of phenotypic characteristics. Predictions of common expressions between diseases are uninteresting since expressions are either observed between diseases or they are not, so practically speaking the value of phenotypic predictions is negligible. Nonetheless, holding out phenotypic links and subsequently predicting their presence is equally instructive for the purposes of predictor evaluation.

`hepth-cite` and `hepth-collab` are formed in exactly the same way as `dblp-cite` and `dblp-collab` respectively. The raw data for these networks is a set of publications in theoretical high-energy physics. In particular, we used a data set post-processed by the Knowledge Discovery Lab at the University of Massachusetts for use in [16] rather than the original 2003 KDD Cup competition data set. This form of the data set offers advantages in data quality and entity consolidation and disambiguation.

The `huddle` data set from [20] is transaction data gathered at a convenience store on the University of Notre Dame campus. The data was collected from June 2004 to February 2007. Products are represented by nodes, and products purchased together in the same transaction are represented by undirected links.

The `patents-collab` data set is constructed from the data at the National Bureau of Economic Research. Nodes represent authors of patents and undirected links are formed between authors who work together on the same patent.

4.2 Experimental Setup

To run our experiments, we integrated VCP with the LPmade link prediction software [14]. LPmade uses a GNU `make` architecture to automate the steps necessary to perform supervised link prediction. This integration will allow those interested in VCP for link prediction and other purposes to test it on their networks easily.

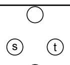
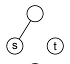
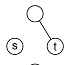
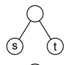
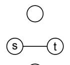
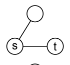
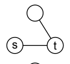
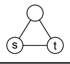
We compare link prediction output against representatives from different predictor families established as strong by prevailing literature [13]. The unsupervised selections include the Adamic/Adar common neighbors predictor [1], the Katz path-based predictor [11], and the preferential attachment model [2, 18]. We also compare against the HPLP supervised link prediction framework contributed by [15] including the PropFlow feature. HPLP combines simple topological information such as node degree and common link predictors into a bagged random forests classification framework with undersampling, a framework that the authors showed works well.

When performing classification using VCPs, we opted for the bagged [3] random subspaces [9] implementation from WEKA 3.5 [22]. This classification scheme offers significantly lower peak memory requirements than random forests while simultaneously providing the potential to handle feature redundancy [9]. We considered presenting results with HPLP also using random subspaces, but we determined that random subspaces produced decreased or comparable performance to the original reference implementation, so we present HPLP results unmodified using random forests [4].

We used the default values from HPLP of 10 bags of 10 random forest trees, 10 bags of 10 random subspaces for VCP classifiers, and training set undersampling to 25% positive class prevalence in training. We did not change the size or distribution of the testing data. For undirected networks, we resolve $f(v_s, v_t) \neq f(v_t, v_s)$, by computing the arithmetic mean to serve as the final prediction output. By default, LPmade includes features that consider edge weights such as the sum of incoming and outgoing link weights, and PropFlow inherently considers edge weights. We are interested in the comparative prediction performance of the link structure alone, so we ran all predictors on the networks disregarding edge weights. There are many different ways to assign edge weights to all the networks here, and the particular choice of edge weight and the precise decision about how to incorporate it into the VCPs would distract from the study.

Computing and evaluating predictions for all possible links on large, sparse networks with any prediction method is infeasible for multiple computational reasons including time and storage capacity. Link prediction within a two-hop geodesic distance provides much greater baseline precision in many networks [15, 21], so effectively predicting links within this set offers a strong indicator of reasonable deployment performance. For all compared prediction methods, we

Table 4: The distributional divergence of highly versus lowly ranked links as output from Adamic/Adar on the sms network.

Address	Element	Distance
0		0.006
1		0.091
2		0.052
3		0.063
4		0.006
5		0.116
6		0.051
7		0.106

restricted the prediction task by distance and only considered performance comparisons for potential links spanning two hops within the training data due to their higher prior probability of formation and computational feasibility.

Reciprocity is an important consideration for link formation in directed networks, so when performing undirected VCP computations on directed networks, we deviate slightly from the definitions above to consider existing reciprocal links as a different relation type and accordingly double the width of the VCP feature space to include elements with and without the reciprocal link.

4.3 Link Analysis

VCPs can assist with a variety of functions regarding link analysis, and post hoc analysis of link prediction output is an interesting example. We start with the performance of the Adamic/Adar predictor in the `sms` network. As we show in Table 5, it achieves AUROC performance of 0.642 and AUPR performance of 0.009410. It may be helpful to us as modelers to understand better why Adamic/Adar fails. We can do this by looking at other simple characteristics of the graph such as degree, centrality measures, or clustering coefficient, but VCPs offer a fine-grained and informative view of links as they are embedded in the broader topology.

We select the Adamic/Adar predictor and first extract the positives from the top 10 million predictions and place them in one set. We place all remaining positives in a second set. For the positives in each of these sets, we can very quickly compute the VCPs of our choice. For simplicity in the demonstration, we choose undirected VCP^{3,1}. Since `sms` is a directed network, we extend VCP^{3,1} to include reciprocal edges between v_i and v_j if they exist. This procedure provides two multi-column distributions of corresponding columns. One logical first step is to compute the distributional divergences of these columns. The distributions are highly skewed, so we use Hellinger distance [10], a non-parametric measure of divergence ranging from 0 to $\sqrt{2}$. The distances are shown in Table 4.

We select the most divergent element, the fifth, and examine the distribution of highly ranked and lowly ranked

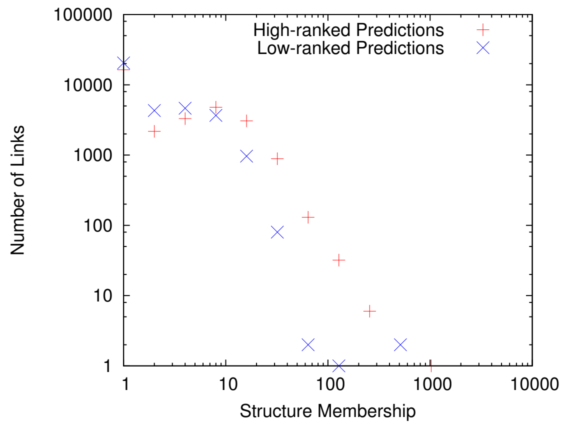


Figure 5: Distributional comparison of extended VCP^{3,1}(5) membership for highly ranked and lowly ranked Adamic/Adar prediction outputs.

Adamic/Adar outputs more closely in Figure 5. The fifth element is one in which a reciprocal link precedes the target link in the prediction. The distributions indicate that highly ranked predictions in Adamic/Adar tend to have more connected source vertices than lowly ranked predictions. Since having many neighbors in common tends to follow from simply having many neighbors, this is not surprising, but the greater dissimilarities of elements 1, 5, and 7 and lesser dissimilarities of 6 and 2 suggest that the connectedness of the link initiator may be more significant than that of the receiver. Adamic/Adar as a model fails to sufficiently separate links containing low-degree source vertices in this network.

In this particular case, we could have obtained the same information by examining the degree distributions of the two sets, but 4-vertex VCPs offer much more distinctive structural information with their greater complexity. This is only one of many ways to perform post hoc link prediction analysis that focuses on the causes of type 2 error in prediction output. Similar analysis could be applied to analyze type 1 error in an attempt to increase precision. Many more powerful and imaginative variations on these techniques apply to link analysis and clustering in general.

4.4 Prediction Performance

The area under the receiver operating characteristic curve (AUROC) can be deceptive in scenarios with extreme imbalance [8] and area under the precision-recall curve (AUPR) exhibits higher sensitivity in the same scenarios [7]. We will provide results for those interested in traditional AUROC, but we will also present AUPR results and will mainly restrict our analysis to those results. Table 5 shows the comparative AUROC and AUPR performance of Adamic/Adar, Katz, preferential attachment, HPLP, and VCPs in link prediction for potential links spanning a geodesic distance of two hops.

In general, we expect the information content of VCPs to increase in the left-to-right order presented in the table. Depending on the significance of directedness in the network, the expectation of performance from VCP 3D and VCP 4U may change. We point the reader to `calls`, `dblp-cite`, `dblp-collab`, `disease-g`, `disease-p`, `hepth-cite`, `huddle`, `patents-collab`, and `sms` as conformant examples. We sus-

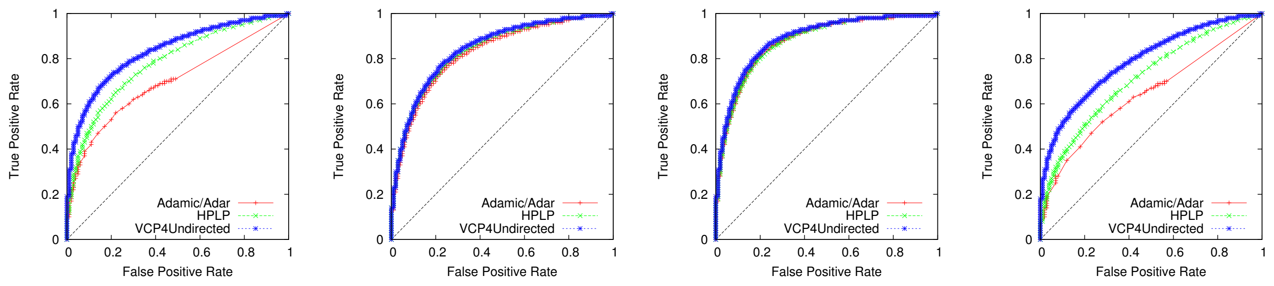
	AA	Katz	PA	HPLP	VCP 3U	VCP 3D	VCP 4U	VCP 4D
calls	0.698	0.641	0.424	0.782	0.802	0.814	0.834	**0.849**
condmat	**0.663**	0.630	0.585	0.588	0.637	-	0.582	-
dblp-cite	0.794	0.791	0.773	0.841	0.830	0.847	0.843	**0.868**
dblp-collab	**0.697**	0.623	0.523	0.691	0.640	-	0.695	-
disease-g	0.930	0.907	0.820	**0.970**	0.923	-	0.964	-
disease-p	0.898	0.920	0.932	0.922	0.939	-	**0.951**	-
hepth-cite	0.826	0.794	0.766	0.838	0.836	0.846	0.845	**0.851**
hepth-collab	0.606	0.619	0.547	0.592	0.598	-	**0.622**	-
huddle	0.879	0.875	0.875	0.877	0.881	-	**0.888**	-
patents-collab	0.793	0.671	0.532	0.800	0.680	-	**0.816**	-
sms	0.642	0.581	0.472	0.714	0.735	0.730	0.791	**0.802**

(a) AUROC

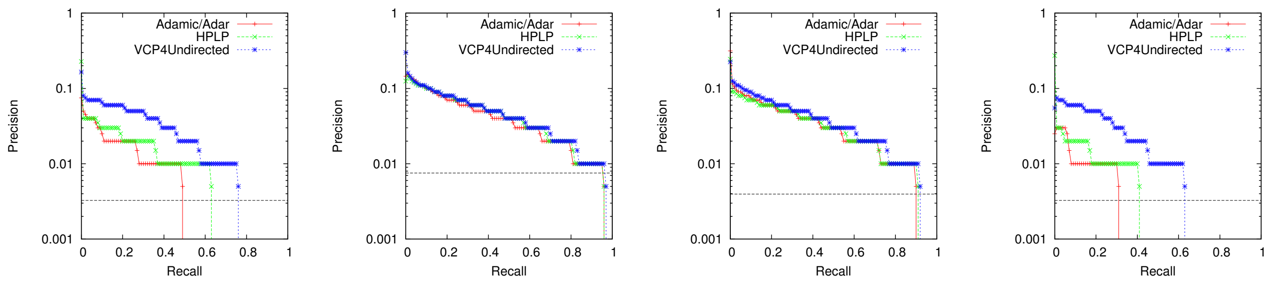
	AA	Katz	PA	HPLP	VCP 3U	VCP 3D	VCP 4U	VCP 4D
calls	0.000505	0.011465	0.000092	0.018005	0.031655	0.033091	0.033533	**0.035127**
condmat	0.000195	0.000183	0.000177	0.007763	**0.011917**	-	0.008589	-
dblp-cite	0.000314	0.000246	0.000234	0.016030	0.009207	0.015265	0.011427	**0.018137**
dblp-collab	0.008777	0.006723	0.003251	0.007772	0.007152	-	**0.009410**	-
disease-g	0.221299	0.193863	0.061694	**0.466716**	0.155165	-	0.444153	-
disease-p	0.629516	**0.676419**	0.673601	0.390074	0.552765	-	0.633316	-
hepth-cite	0.003967	0.003784	0.003225	0.054846	0.046140	0.059245	0.056244	**0.063387**
hepth-collab	0.008563	**0.009328**	0.005060	0.006123	0.007197	-	0.007156	-
huddle	0.000790	0.000746	0.000745	0.039914	0.039394	-	**0.046803**	-
patents-collab	0.006962	0.005678	0.001684	0.006735	0.005564	-	**0.007709**	-
sms	0.009410	0.009164	0.002986	0.011594	0.025206	0.026063	0.027073	**0.028201**

(b) AUPR

Table 5: Comparative performance for Adamic/Adar (AA), Katz, preferential attachment (PA), HPLP, and VCP. For VCP, U indicates that directionality is ignored and D indicates that it is considered.



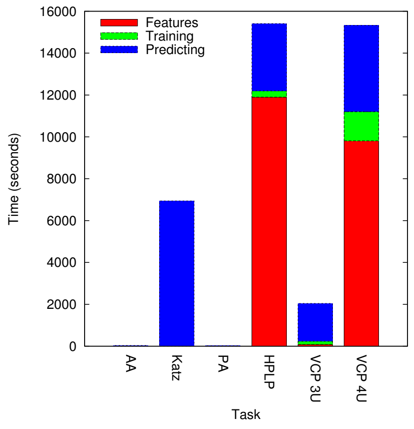
(a) AUROC



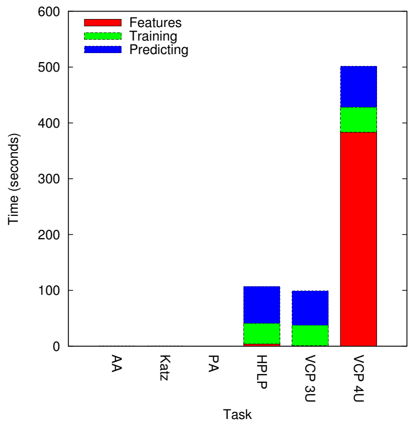
(i) calls (ii) hepth-cite (iii) huddle (iv) sms

(b) AUPR

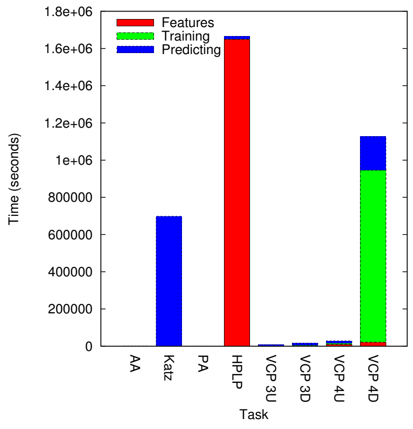
Figure 6: ROC and precision-recall curves for selected networks.



(a) dblp-collab



(b) disease-g



(c) sms

Figure 7: Timing analysis for three different networks. Network size information is available in Table 3.

pect that the exceptions indicate cases in which the classification algorithm was either overfitting the training set or failed to create a sufficiently optimized model in the high-dimensional space.

In 7 of the 11 networks, VCP classification offers superior AUPR performance. In a slightly different selection of 7 networks, it offers superior AUROC performance. In some of the cases in which VCP offers the best performance, the differences are quite wide. In the `sms` network it offers an AUPR that is 2.3 times as high as the best competitor. In the `condmat` network, it offers AUPR 1.53 times the nearest competitor. In two of the networks in which VCP classification does not provide the best performance, HPLP does. As an interesting side note, when weights are removed as they were to obtain these results, HPLP does not always outperform the unsupervised predictors.

Figure 6 shows a closer look at the performance differences. The black dashed line represents the baseline performance of a random predictor. Across all the selected networks, VCP maintains high precision longer at increasing values of recall. This is especially important in link prediction where high precisions are so difficult to achieve.

Despite the strong and competitive performance that the VCP method of supervised prediction exhibits, it is not our intent to present the most effective possible classification scheme. Our experiences with random forests, random subspaces, and other classification techniques suggest that the potential for improvement through feature selection and alternative classification algorithms is high. Another option for potential improvement is to concatenate $\text{VCP}^{3,r}$ and $\text{VCP}^{4,r}$ into a single feature vector. VCPs contain much information, and the task is simply to determine how best to leverage it to achieve whatever goals are desired.

4.5 Timing

We used two different types of machines for timing. All feature computation and VCP generation was performed serially on a quad-core Opteron 2218 running at 2.6 GHz with 1 MB cache and 4 GB of main memory. Some classification runs required more than 4 GB of memory with the specified training set undersampling and algorithm parameters, so all WEKA classification was performed on a dual quad-core Xeon E5620 running at 2.4 GHz with 12 MB cache and 32 GB of main memory. To some degree, timings are implementation dependent, and though the implementations of predictors, feature computations, and VCPs are not naïve, we cannot claim that they are fully optimized. Figure 7 shows the results.

Adamic/Adar, $O\left(\frac{|E|}{|V|}\right)$ per prediction, and preferential attachment, $O(1)$ per prediction, perform very few operations to generate their output. They are so inexpensive to compute that they are invisible within the same scale as Katz and the supervised prediction methods for all three networks in Figure 7. We note that for all three networks the total time to perform supervised link prediction with VCPs is often less than that necessary for HPLP. Most of this is due to the expense of the Katz feature, which involves finding paths up to length 5 with the aid of memoization in our implementation. Based on these results, Adamic/Adar is clearly an effective and inexpensive option for a wide variety of networks, but VCP offers significant potential for performance enhancement.

Perhaps the most interesting conclusion lies in the inconsistency of the results in terms of the breakdown of time requirements for different components. Timing is related to the interplay between the specific algorithms involved, the local densities or global density of the graph, and the raw size of the graph in terms of the number of vertices and edges. Whether the bulk of time is consumed in feature generation, VCP computation, training, or testing varies greatly across networks as does the total time for any particular method.

The VCP implementations provided are slightly limited in efficiency because of the graph implementation to which they are tied in LPmade. With a more amenable supporting graph implementation and slight changes to the selection of data structures, we expect that it would be possible to decrease the running time of the VCP vector computation itself by a factor of at least 2. Nonetheless, the computation of even $\text{VCP}^{4,2}$ is competitive in terms of running time with much simpler and less effective path-based prediction methods.

The results in Figure 7 show that VCP is more efficient from a cost-performance standpoint than classification based on computing and combining simpler unsupervised predictors. Further, VCP computations are naturally parallel, and the extended times for the `sms` network include computing VCPs for tens of millions of vertex pairs. The extended feature vector of VCP^{4,2} greatly increases training time, but feature selection or the application of different training algorithms or parameters could reduce this greatly, and training is parallelizable across bags.

5. CONCLUSIONS

VCP is a new method of link analysis with solid theoretical roots. We presented evidence of its utility in some applications here, but there are many possible applications. It is useful for post hoc analysis of classification output, comparative analysis of network link structure, and it competes effectively with an existing recently published method, HPLP, often outperforming it by wide margins. In well-established networks with past observational data, VCP can serve as a sensitive change detection mechanism for tracking the evolving link formation process. In addition to link prediction and link analysis for the purpose of network growth modeling, VCP can be used for link or vertex pair clustering. Its ability to handle multiple relations naturally extends its utility into many domains and offers an alternative to the practice of combining or discarding edge types or edge directionality.

The VCP computations for the directed and undirected variants of both VCP^{3,1} and VCP^{4,1} are integrated into the LPmade link prediction framework. The LPmade branch containing the software is available at `http://mloss.org/software/view/307/`. Most of the data sets are publicly available elsewhere, but we have also published all public data sets to `http://nd.edu/~rlichten/vcp` so that our experiments can be repeated with the same longitudinal thresholds and thus the same network saturation. We have provided code to perform VCP subgraph-to-element mappings at the same location.

6. ACKNOWLEDGMENTS

Research was sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 and in part by the National Science Foundation (NSF) under Grant BCS-0826958. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

7. REFERENCES

[1] L. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2001.

[2] A.-L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaboration. *Physica A*, 311(3-4):590–614, 2002.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] R. S. Burt. *Structural holes: The social structure of competition.* Harvard University Press, 1995.

[6] D. Davis, R. N. Lichtenwalter, and N. V. Chawla. Multi-relational link prediction in heterogeneous information networks. In *Proc. of the Intl. Conf. on Adv. in Social Networks Analysis and Mining*, 2011.

[7] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proc. of the 23rd Intl. Conf. on Machine Learning*, pages 233–240. 2006.

[8] D. Hand. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, 77(1):103–123, 2009.

[9] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[10] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Trans. on Communications Technology*, 15(1):52–60, 1967.

[11] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[12] D. L. Kreher and D. R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*, chapter 7, pages 253–264. CRC Press, 1 edition, 1999.

[13] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[14] R. N. Lichtenwalter and N. V. Chawla. Lpmade: Link prediction made easy. *Journal of Machine Learning Research*, 12:2489–2492, 2011.

[15] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *Proc. of the 16th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 243–252, New York, NY, USA, 2010.

[16] A. McGovern, L. Friedland, M. Hay, B. Gallagher, A. Fast, J. Neville, and D. Jensen. Exploiting relational structure to understand publication patterns in high-energy physics. *ACM SIGKDD Explorations Newsletter*, 5(2):165–172, 2003.

[17] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824, 2002.

[18] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review Letters E*, 64, 2001.

[19] N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):177–183, 2007.

[20] T. Raeder, O. Lizardo, D. Hachen, and N. V. Chawla. Predictors of short-term decay of cell phone contacts in a large-scale communication network. *Social Networks*, 33(4):245–257, 2011.

[21] S. Scellato, A. Noulas, and C. Mascolo. Exploiting place features in link prediction on location-based social networks. In *Proc. of the ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2011.

[22] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, San Francisco, California, USA, second edition, 2005.