SAND2007-5820P

Clustering Attributed Relational Graphs for Information Organization CARGIO: FY07 First Quarter Report

1 Summary of Milestone Progress

1a: Milestone: *Literature review.* We will become cognizant of current state-of-the-art algorithms for clustering univariate and attributed relational graphs.

Percent Complete: 100%

Comments: Complete. Happily, the premise and research plan of the initial proposal still seems suitable.

1b: Milestone: *Acquire test data.* We will obtain a number of directories containing multiple projects from different users; both DART and non-DART data will be included.

Percent Complete: 90 or 100%; see Section 2.2.

Revised Completion Date: April 1, 2007.

Comments: We have three file hierarchy data sets (two pertinent to DART, one general), 78.3 Gbytes in size, with a total of 159,785 files.

1c: Milestone: *Data familiarization and ontology formation.* We will become familiar with the test data and develop a small but extensible initial ontology for the graph representation.

Percent Complete: 100%

Comments: The initial ontology provides for simple nodes and edges, both with multiple attributes, and unconstrained connections.

1d: Milestone: *Begin to groundtruth test data.* We will develop an initial list of projects and assign tentative file memberships. Because of its qualitative nature, we expect the groundtruthing of the data to be an interative process.

Percent Complete: 0%

Revised Completion Date: April 1, 2007.

Comments: The test data was not obtained until late in the first quarter. This task will be pushed into the next quarter, and folded into milestone 2b, "Refine groundtruthing of test data".

1e: Milestone: *Decide upon candidates for performance metrics.* These candidates will be suggested by our literature review, and will be used to evaluate the accuracy of the methods we develop.

Percent Complete: 100%.

Comments: A small subset of appropriate measures have been culled from the literature.

1f: Milestone: *Formulate a detailed and prioritized list of software features necessary for manipulating and studying attributed relational graphs.*

Percent Complete: 100%

Comments: Current list suffices to begin implementation; we anticipate refining the list as part of milestone 2e.

2 Milestone Details

2.1 Literature review

Goal: *We will become cognizant of current state-of-the-art algorithms for clustering univariate and attributed relational graphs.*

2.1.1 Univariate Graph Clustering

We have discovered, but not yet thoroughly explored, an abundance of univariate graph clustering algorithms [18, 34, 7, 37, 33, 38, 30, 5, 12, 8, 11]. A terse overview:

- An attempt to improve upon the computational speed of spectral based clustering algorithms by using a general cut objective function and a multilevel clustering approach [7].
- The development of a Markov clustering (MCL) method based upon the property that a random walk through a graph will tend to get stuck in a cluster [37].
- A divisive method for finding clusters by sequentially deleting edges, along with an objective metric for determining the number of clusters (i.e. when to halt their algorithm) [30].
- An enhancement of the above divisive method that shows how its optimization can be reformulated as a spectral relaxation problem [38].
- An algorithm called Geometric MST (Minimum Spanning Tree) Cluster (GMC) [5], which is compared[1] with both MCL [37] and Iterative Conductance Cutting (ICC) [5].
- A technique for clustering graphs that are changing over time, with nodes being added and deleted[12].
- A divisive hierarchical algorithm that uses spectral clustering and a normalized cut objective function [33].
- A Monte Carlo randomized algorithm which finds the minimum cut of an undirected graph with edge weights [18].
- A k-clustering agglomerative algorithm for unweighted graphs that begins with every node being a cluster and terminates when the clustering has stabilized [34].

We will revisit, and prioritize, the above list of methods when we address univariate graph methods in the second year of this CSRF project.

2.1.2 Attributed Relational Graph Clustering

As for clustering the nodes of attributed relational graphs, we found, as suspected, scant literature. All methods found first develop a mechanism for reducing an ARG to a weighted graph and then cluster the weighted graph using standard techniques [32, 6, 17, 28, 29, 3, 4]. (This is precisely the approach we plan to explore in the second year of this project.)

There has been much work done on software clustering [2, 22, 21, 22, 27]. This seems similar to our interests; the primary difference is that software clustering is trying to divide up the functionality of a single software package into modules. However, the nodes are files and how they relate are (sometimes multi-attributed) edges. One example is in [32], where they begin by forming an ARG describing various file relationships; this is similar to one aspect of our ontology. The ARG is then transformed to a component graph (we believe this means a weighted graph where nodes may be a group of files). This component graph is then clustered using standard univariate graph clustering techniques. In [28, 29], an ARG is converted

[1] Oddly, the only reference we can find to ICC is in the same paper that describes GMC

to a weighted graph by combining attribute and link information into a single edge weight. In [28] three univariate clustering algorithms are explored: Karger Min-Cut [18], an extension of MajorClust [34], and a spectral clustering approach based on [33]. In [29] clusters are found using spectral clustering techniques on the resultant adjacency matrix. The ARG is updated and the process repeated.

Papers by [6, 17, 3, 4] all address the same problem: author disambiguation. Here each node represents a variant on an author's name. One approach is to first create an ARG of all possible relations [6, 17]. Then they group the nodes into *virtual connected subgraphs* where each subgraph is composed of a list of nodes linked by ambiguity. That is, all other nodes outside this list are definitely not related to the nodes within the list. Once these subgraphs are created, clustering is performed on each subgraph to determine the distinct authors in this set. In [6] the subgraph (which is also an ARG) is reduced to a weighted graph by choosing a *connection strength model* that is used to compute edge strength; it is assumed that a domain expert assigns the corresponding weights. In [17] this effort is extended by using data mining techniques to choose the appropriate weights for the connection strength model.

Authors in [3] appear to follow a similar framework but begin by clustering author names into entities that are *obviously* the same person; this forms the initial clustering of duplicate authors. They also create an alternate partition G of the set of authors; they do not state how this is done, but it seems to be based upon paper content, i.e. topic-based document clustering. Distance is then defined in terms of a convex combination of a distance measure between attributed author nodes, $d_{attr}(r_i, r_j)$ and a distance measure between the groups in G that contain those authors and their duplicates, $d_{group}(G(r_i), G(r_j)$, i.e.

$$d(r_i, r_j) = (1 - \alpha)d_{attr}(r_i, r_j) + d_{group}(G(r_i), G(r_j)).$$

They do not explicitly define a measure for computing $d_{attr}(r_i, r_j)$ but cite examples of software packages that can do so [9]. Oddly, though Elfeky et al. [9] appear to be describing the software package, *TAILOR: A Record Linkage Toolbox*, there does not appear to be a website associated with this package. For a survey of other record disambiguation approaches see [10].

Finally, we did find several examples of research into a slightly related task, involving analysis of multirelational data. In [19], an algorithm is developed for simultaneously clustering multiple sets of related data, extending properties of the spectral clustering algorithm by collective factorization of related matrices. A larger body of work exists for the clustering of a set of separate ARGs [16, 14, 31], essentially via graph matching.

2.2 Acquire test data

Goal: *We will obtain a number of directories containing multiple projects from different users; both DART and non-DART data will be included.*

We have obtained three file directories trees to serve as our initial data sets:

- **tdkostk**: Combined Shasta and AMECH LAN home directories from Tim Kostka, who was working on structural mechanics analysis. Thus, this directory tree will be suitable for the DART application of the eventual CARGIO methods. This data set contains 4,625 files comprising a total of 2.3 Gbytes.
- **sgoff:** Combined Shasta and AMECH LAN home directories from Shannon Goff, who was working on structural mechanics analysis. Again, a directory tree suitable for the DART application. This data set contains 4620 files comprising a total of 36 Gbytes.
- **wkoegl:** The home directory of Wendy Koegler Doyle. This directory reflects six years of work on a wide variety of projects, none of them primarily structural mechanics analysis. Thus this directory tree is more suitable for investigation of the non-DART, more law-enforcement oriented CARGIO application. This data set contains 150,540 files comprising a total of 30 Gbytes.

Further, in order to explore the possibility that the CARGIO methods might be applicable to Nuclear Weapons Information Environment (NWIE) concerns, we are currently seeking to acquire a subset of the Web File Share (WFS) files. We are in discussion with the WFS project lead, Beth Moser, about the

advisability of, and practical issues around, this idea. If a WFS data set eventually proves feasible, this milestone is currently 90% complete. If not, we are done for now.

2.3 Data familiarization and ontology formation

Goal: *We will become familiar with the test data and develop a small but extensible initial ontology for the graph representation.*

2.3.1 Node attributes

Each node represents a single "file", in the general sense, including directories and such. Each node may have a series of attributes, each attribute taking a value of a specified data type.

The initial set of node attributes is listed below. Note that there is scant computational cost associated with *node* attributes. Note also that there are properties of the files used in computing edges (for instance, all of the words in the file) that are referenced when the graph is built but not stored in the node proper.

name: *filename*
datatype: string
comments: Full pathname of file.

name: *owner*
datatype: string
comments: Denotes owner of a file.

name: *permissions*
datatype: string
comments: Read/write permission settings for file.

name: *time_created*
datatype: int
comments: Date of file creation.

name: *time_modified*
datatype: int
comments: Date of last file modification.

name: *time_access*
datatype: int
comments: Date of last file access.

name: *group*
datatype: string
comments: Group file belongs to.

name: *readable*
datatype: bool
comments: True if file is ascii text, false if binary.

name: *type*
datatype: string
comments: Could be one of the following: directory, executable, library, data, image, source, compiled, document, or other. The category *compiled* is used to signify compiled files that are not also executables, i.e. foo.o.

name: *cluster_<name>*
datatype: string
comments: <name> is specified by the user. This value denotes a label specifying to which cluster this node belongs. Each time a cluster algorithm is run, a new labeling can be added to the node, storing the

new cluster values. Thus later we can evaluate different sets of labelings for the same graph using the same file. Similarly, we can store multiple "truth" labelings, to reflect, for instance, the initial assignment, a revised assignment based on feedback from users, and so on.

2.3.2 Edge attributes

Each edge represents a relationship between two files, including directories and such. Each edge may have a series of attributes, each such attribute taking a value over $\mathbb{R}$.

An initial set of node attributes is listed below. Note that there *is* notable computational cost associated with each *edge* attribute implemented. Each new edge attribute not only creates an additional "layer" in, for instance, a tensor model of the graph, but it also creates a large set of additional edges. Therefore we are likely to initially implement only a small subset of these edge attributes, adding more as the application demands and as the computational cost becomes clearer.

name: *time_delta*
comments: $\min_{\mathrm{T}\in(\mathrm{created,modified,accessed})} |\mathrm{time\_T}[N_1] - \mathrm{time\_T}[N_2]|$. Denotes difference in seconds between time stamps for the files linked by this edge.

name: *ancestry*
comments: Denotes minimum number of directories one must traverse upwards to travel from location of file N_1 to location of file N_2. No edge is formed if there is no ancestry relation.

name: *name_match*
comments: Measures how well the filename of N_1 matches the filename of N_2.

name: *text_match*
comments: Measures how well the text contained in file N_1 matches that of file N_2.

name: *generates*
comments: True if N_1 is used to generate N_2. Thus library files will always be edge sinks for their corresponding source files. Still to be decided: suppose main.c generates main.o generates a.out. Does main.o *and* main.c link to a.out? If not, what if main.o is removed? It seems main.c and a.out should still be linked. Furthermore, what about generator helpers, such as Makefiles, configure scripts, etc?

name: *namedrop*
comments: File N_1 and file N_2 are listed together in the same text file (minus possibly suffixes) in a same or parent directory with file ancestry distance less than k, a user-defined parameter. This is to account for files described in parent directory README, Content, Makefile.am, CVS/Entries, type files.

name: *link*
comments: Signifies that N_1 is a symbolic link to N_2.

name: *external*
comments: Denotes probability, on $[0, 1]$, that two nodes are in same project, as set, interactively, by a user. This is the only edge attribute currently considered whose value is not inferred solely from the data. The point is to provide a mechanism for capturing analyst information about the data.

2.4 Begin to groundtruth test data

Goal: *We will develop an initial list of projects and assign tentative file memberships. Because of its qualitative nature, we expect the groundtruthing of the data to be an iterative process.*

This task has not yet begun. It was dependent on both tasks 1b (obtain data) and 1e (determine cluster validation metrics), and 1b was not complete until late in the first quarter. This was not unanticipated, which is why a second quarter milestone (2b, refine groundtruthing) was in the original plan.

We have, however, refined our ideas on how to best approach a process which is both technically tricky and practically tedious. For each test data set, we will combine rough but insightful human estimates of

the clusters with detailed but likely less accurate computer generated estimates, hoping thereby to achieve accurate clusters with a reasonable amount of human effort. In more detail:

- We will first determine a rough guess at the correct groundtruth by crude, hand derivation of initial clusters $\mathcal{G}_1$.
- We will then apply internal cluster validation to $\mathcal{G}_1$ and modify the partition accordingly, resulting in modified clusters $\mathcal{G}_2$. By "internal" cluster validation we mean those methods that assess the fitness of clusters in and of themselves, without reference to a "correct" clustering.
- Once we have any initial clustering algorithm implemented, we will use it to generate clusters $\mathcal{G}_3$.
- Then we will again apply internal validation techniques to $\mathcal{G}_3$, and modify accordingly to obtain $\mathcal{G}_4$.
- Finally, we will use relative validation techniques, visual inspection, and user feedback to compare and contrast $\mathcal{G}_4$ and $\mathcal{G}_2$, learning from both to create the final "truth" clusters $\mathcal{G}_f$.

We may then begin to use $\mathcal{G}_f$ to formulate and evaluate different performance metrics that will be used to assess the quality of different clustering algorithms and algorithmic parameter settings.

2.5 Candidates for performance metric

Goal: *Form list of candidates suggested by our literature review that will be used to evaluate the accuracy of the methods we develop.*

We have found several interesting papers [23, 24, 25, 13, 15, 35, 36, 39] that deal extensively with cluster validation techniques and that suggest several good possibilities for a performance metrics. We will evaluate CARGIO clusters via the criteria of "stability", "extremity", and "authoritativeness", as suggested by [39]:

- *Stability:* does the clustering change dramatically if only a few files are changed? Stability is clearly important as the addition of a small number of files should have the effect that they are primarily absorbed by one or more preexisting clusters. However, if the resulting clustering changed dramatically for all files, then our algorithm is likely not giving us the correct information. One possibility for testing this is to use snap shots of data sets taken from the same user over a period of time.

  A mechanism for quantifying stability was developed by Wu [39]. They first define an ordinal measure ranking two sequences of real numbers S_i, S_j :

  $$Below(S_i, S_j) = \frac{|\{n \mid S_i[n] < S_j[n], 1 \leq n \leq |S_i|\}|}{|S_i|}$$

  and

  $$Below(S_i) = \sum_{j=1}^{K} Below(S_i, S_j).$$

  Intuitively one may think of $Below(\cdot, \cdot)$ and $Below(\cdot)$ as a measurement comparing a sequence of performance scores on a suite of test scenarios between multiple clustering algorithms. That is to say, we wish to compare multiple algorithms on multiple test sets. For examples, suppose

  $$S_a = \{2, 3, 8, 8\},\ S_b = \{3, 1, 9, 6\},\ \text{and}\ S_c = \{4, 4, 7, 7\}$$

  are the performance scores for clustering algorithms a, b, and c respectively on 4 different tests sets. We see on the first test set that algorithm a, b, and c scored a 2, 3, and 4, respectively. Thus for the first test set, algorithm a outperforms algorithms c and b (assuming lower scores are preferable).

However, with many test sets and many algorithms this could quickly become confusing. So one possibility is to use the *Below* measurement developed by Wu [39]. So for the above examples

$$Below(S_a, S_b) = \frac{(2<3)+(3<1)+(7<9)+(7<6)}{4} = \frac{1+0+1+0}{4} = .5$$

$$Below(S_a, S_c) = \frac{(2<4)+(3<4)+(7<2)+(7<2)}{4} = \frac{1+1+0+0}{4} = .5$$

and

$$Below(S_a) = Below(S_a, S_a) + Below(S_a, S_b) + Below(S_a, S_c) = 0 + .5 + .5 = 1.$$

Similar calculations show that $Below(S_b) = 1.25, Below(S_c) = 0.75$, indicating that B is the best of the three algorithms, as it has the highest "below" score.

In the context of stability measurements, we can think of a sequence S_i of dissimilarity measures for a given algorithm between clusterings for set of sequentially perturbed test sets. Since each test set is similar to the previous, we would hope that the resulting clusters would also be similar. So ideally, S_i would have all small numbers. If another sequence, S_j for an alternate algorithm exists, we would say algorithm i outperforms algorithm j if S_i is "below" S_j. Note, we may wish to modify such a measurement to account for the difference in magnitude between two measurements.

Suppose then that we are given a sequence of clusterings $P = \{P_k\}_{k=1}^{T}$ where P_k denotes a partitioning of the graph (not an individual cluster). Then given any similarity measure between clusterings, we can form a sequence $\{S_k\}_{k=1}^{T-1}$ where S_k denotes the similarity between clusters P_k and P_{k+1}.

Now suppose we have K distinct clustering *algorithms*, $\{A_k\}_{k=1}^{K}$. Each algorithm is used to generate a sequence of T clusterings and a corresponding similarity list $S^{(k)} = \{S_j^{(k)}\}_{j=1}^{T-1}$. Then the score for algorithm k, $1 \le k \le K$ is given by

$$Score(A_k, P) = Below(S^{(k)}) = \sum_{\ell=1}^{K} Below(S^{(k)}, S^{(\ell)}).$$

- *Extremity of clusters:* are the resulting clusters large enough to provide big picture information, but small enough to understand the major project themes?. An algorithm that creates many very small clusters or a small number of very large clusters is not providing information that aids human insight. I.e., possible worst case scenarios would be if every file were deemed its own project or if the entire data sets was considered a single project.

  A simple mechanism for quantifying extremity is NED (non-extreme distribution) [39]. Its application requires the user specify upper and lower thresholds for cluster sizes; any cluster outside those bounds is "extreme". Then NED is the ratio of the number of elements in *non-extreme* clusters to the total number of files. Clearly this is optimized at 1, signifying no extreme clusters, and is 0 if all clusters are extreme.

  The extremity of clusters for an algorithm A_i can then be comparatively scored using the following measure:

$$Score_{\mathrm{NED}}(A_i) = Above(\mathrm{NED}(A_i)),$$

  where

$$Score_{\mathrm{NED}}(A_i, A_j) = Above(\mathrm{NED}(A_i), \mathrm{NED}(A_j))$$

  with *Above* having an analogous definition to that of *Below*. Here $NED(A_i)$ denotes the sequence of NED ratios for the set of clusterings $\{P_k^{(i)}\}_{k=1}^{T}$ generated by algorithm A_i for the T distinct data sets.

- *Authoritativeness:* how well do the resulting clusters agree with the groundtruthed data sets? We intend to assess the quality of a handful of performance metrics that compare proposed clusters to the "true" groundtruth clusters.
  - One method is to run the *Kuhn-Munkres algorithm* [20] to obtain the optimal match between two clusterings. This essentially finds the permutation which makes the diagonal of the resulting (possible zero padded) square cluster contingency matrix as close to diagonally dominant as possible. This is computationally demanding, but does sometimes find a better match of proposed to true clusters than straightforward greedy matching.
    Then to form a metric we would subtract the sum of diagonals from twice the total number of files.
  - The following performance metric suggested by van Dongen [35] may be useful in that it is a simply computed metric on the set of partitions:
    $$\mathcal{D}(C, C') = 2n - \sum_{k} \max_{k'} |C_k \cap C'_{k'}| - \sum_{k'} \max_{k} |C_k \cap C'_{k'}|.$$
    Note that
    $$\mathcal{D}(C, C') = \mathcal{D}(C, C \cap C') + \mathcal{D}(C', C \cap C').$$
    Dongen [35] asserts that his distance measure denotes the number of node substitutions needed to convert one partition into another. Perhaps there is an easy way to see why this is true; this sounds suspiciously like the Hamming distance however.
  - The following *variation of information* metric was developed by Meila [23]:
    $$V(C, C') = H(C) + H(C') - 2I(C, C')$$
    where
    $$H(C) = -\sum_{k=1}^{K} \frac{n_k}{n} \log\left(\frac{n_k}{n}\right)$$
    and
    $$I(C, C') = \sum_{k=1}^{K} \sum_{k'=1}^{K'} \frac{|C_k \cap C'_{k'}|}{n} \log\left(\frac{|C_k \cap C'_{k'}|}{n}\right)$$
    This metric can be written as the sum of two positive terms:
    $$V(C, C') = \big[H(C) - I(C, C')\big] + \big[H(C') - I(C, C')\big]$$
    The two terms represent conditional entropies and can be thought of as the amount of information that is gained by moving C to C' and the amount of information that would be lost if C' were moved to C.
  - Another well known metric listed in [35], referenced from [26], is the equivalence mismatch coefficient:
    $$emc(C, C') = \sum_{k=1}^{K} |C_k|^2 + \sum_{k'=1}^{K'} |C'_k|^2 - 2\sum_{k=1}^{K} \sum_{k'=1}^{K'} |C_k \cap C'_{k'}|^2$$
    which is usually scaled by $1/n^2$ to restrict to the interval to $[0, 1]$. This distance has a direct relation to the *Hamming distance*. The Hamming distance can be thought of as the number of substitutions that would be needed to transform the binary characteristic matrix M_{ij} for C to M'_{ij} for C', where
    $$M_{ij} = \begin{cases} 1, & \text{if node } i \text{ and node } j \text{ are in the same partition} \\ 0, & \text{otherwise.} \end{cases}$$

Dongen [35] asserted the benefits of his performance metric over the emc metric using the following example. Suppose that $V = \{1, 2, \ldots, m^2\}$ denotes a set of nodes with $n = m^2$. Let P_1 denote the single partition $\{V\}$ and $P_n = V$ (i.e. every cluster is a singleton). Then consider the matrix

$$M = \begin{pmatrix} 1 & 2 & \cdots & m \\ m+1 & m+2 & \cdots & 2m \\ \vdots & \vdots & & \vdots \\ m^2-m+1 & m^2-m+2 & \cdots & m^2 \end{pmatrix}$$

Then let P_r denote the cluster formed from taking the rows of M and let P_c denote the clusters taking the columns of M. Then we have that

$$d(P_1, P_n) = n - 1 \text{ and } d(P_r, P_c) = 2\sqrt{n}(\sqrt{n} - 1)$$

while

$$emc(P_1, P_n) = n(n-1) \text{ and } emc(P_r, P_c) = 2n(\sqrt{n} - 1).$$

Hence Dongen's performance metric gives (P_1, P_n) a higher score than (P_r, P_c) while emc reverses this order. However, P_1 is a sub-partition of P_n while P_r and P_c are maximally conflicting. This is interesting, but perhaps not overwhelmingly convincing, as it pertains to extreme cases only.

2.6 Determine graph visualization/analysis tools needed

Goal: *Formulate a detailed and prioritized list of software features necessary for manipulating and studying attributed relational graphs.*

Note: the following list is roughly prioritized. As a result, some of the entries are extended or expanded versions of entries earlier in the list.

1. Ability to visualize graph. Clicking on an edge should display edge attributes, clicking on a node should display node attributes.

2. Ability to zoom in and out, rotate, and translate graph using mouse.

3. Ability to display properties of the graph interactively, or based directly on node or edge attributes, using shading, dimming, coloring schemes, edge thickness, and node size.

4. Ability to click on an edge or node and view in a slaved pop-up window the corresponding attributes. Should be able to edit node and edge attributes using this slaved window.

5. Ability to zoom in by outlining with the mouse a rectangular block of nodes.

6. Ability to select a group of nodes by 1) outlining with the mouse a rectangular area, and 2) holding down the alt key and continuing to click on nodes/edges.

7. Ability to select a group of nodes or edges by typing in a list of node or edge attributes.

8. Ability to click on a node group and see in a text box a list of each file name contained in the cluster. These filenames should be hyperlinked to both the slave node window mentioned above and the actual file so that the user can view its contents if readable and view the dump from the file command otherwise:

   - If the file is viewable (pdf, html, text, etc), then the file will pop up with the appropriate viewer.
   - If file is binary, the user will be given the information available from both the ls and file commands.
   - If the file is a library file, then the names of the corresponding files will be included.

- If the file has unknown format, then we should provide an *open with* command.

9. Ability to layout graph based upon cluster labels, i.e. the layout should group node placement so that clustered nodes are together.

10. Ability to place color shades around convex hull of nodes belonging to the same cluster.

11. Ability to manually modify the layout by drag and drop.

12. Ability to switch between multiple automatic layout capabilities.

13. A Matlab-like interface where graph analysis/visualization commands can be typed in as in the software package GUESS [1].

References

[1] E. Adar, *GUESS: A language and interface for graph exploration*, in CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, 2006, ACM Press, pp. 791–800.

[2] N. Anquetil and T. Lethbridge, *File clustering using naming conventions for legacy systems*, in CASCON '97: Proceedings of the 1997 Conference of the Centre for Advanced Studies on Collaborative Research, IBM Press, 1997, p. 2.

[3] I. Bhattacharya and L. Getoor, *Iterative record linkage for cleaning and integration*, in DMKD '04: Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, New York, NY, USA, 2004, ACM Press, pp. 11–18.

[4] ——, *Relational clustering for multi-type entity resolution*, in MRDM '05: Proceedings of the 4th International Workshop on Multi-Relational Mining, New York, NY, USA, 2005, ACM Press, pp. 3–12.

[5] U. Brandes, M. Gaertler, and D. Wagner, *Experiments on graph clustering algorithms*. Lecture Notes in Computer Science, Di Battista and U. Zwick (Eds.):568–579, 2003.

[6] Z. Chen, D. V. Kalashnikov, and S. Mehrotra, *Exploiting relationships for object consolidation*, in Proc. of International ACM SIGMOD Workshop on Information Quality in Information Systems (ACM IQIS 2005), Baltimore, MD, USA, June 17 2005.

[7] I. Dhillon, Y. Guan, and B. Kulis, *A fast kernel-based multilevel algorithm for graph clustering*, in KDD '05: Proceeding of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, New York, NY, USA, 2005, ACM Press, pp. 629–634.

[8] J. Edachery, A. Sen, and F.-J. Brandenburg, *Graph clustering using distance-k cliques.*, in Graph Drawing, Springer-Verlag, 1999, pp. 98–106.

[9] M. Elfeky, V. Verykios, and A. Elmagarmid, *TAILOR: a record linkage toolbox*, in ICDE '02: Proceedings of the 18th International Conference on Data Engineering, IEEE Computer Society, 26 Feb.-1 March 2002, pp. 17–28.

[10] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, *Duplicate record detection: A survey*, IEEE Transactions on Knowledge and Data Engineering, 19 (2007), pp. 1–16.

[11] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, *Graph clustering and minimum cut trees*, Internet Mathematics, 1 (2004), pp. 385–408.

[12] M. Gaertler, R. Görke, D. Wagner, and S. Wagner, *How to cluster evolving graphs*, Tech. Report DELIS-TR-0382, DELIS – Dynamically Evolving, Large-Scale Information Systems, 2006.

[13] A. D. Gordon, *Cluster validation*, in Data Science, Classification, and Related Methods, C. Hayashi, N. Ohsumi, K. Yajima, Y. Tanaka, H. Bock, and Y. Bada, eds., no. IRN 18349225 in Studies in Classification, Data Analysis, and Knowledge Organization, Spring-Verlag, 1998, pp. 22–39.

[14] S. Gunter and H. Bunke, *Self-organizing map for clustering in the graph domain*, Pattern Recogn. Lett., 23 (2002), pp. 405–417.

[15] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, *On clustering validation techniques.*, J. Intell. Inf. Syst., 17 (2001), pp. 107–145.

[16] B. J. Jain and F. Wysotzki, *Central clustering of attributed graphs*, Mach. Learn., 56 (2004), pp. 169–207.

[17] D. V. Kalashnikov and S. Mehrotra, *Domain-independent data cleaning via analysis of entity-relationship graph*, ACM Trans. Database Syst., 31 (2006), pp. 716–767.

[18] D. R. Karger, *Global min-cuts in rnc, and other ramifications of a simple min-out algorithm*, in SODA '93: Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, 1993, Society for Industrial and Applied Mathematics, pp. 21–30.

[19] B. Long, Z. M. Zhang, X. Wu;, and P. S. Yu, *Spectral clustering for multi-type relational data*, in ICML '06: Proceedings of the 23rd International Conference on Machine Learning, New York, NY, USA, 2006, ACM Press, pp. 585–592.

[20] L. Lovász and M. Plummer, *Matching Theory*, Akadémiai Kiadó, North Holland, Budapest, 1998.

[21] C.-H. Lung, M. Zaman, and A. Nandi, *Applications of clustering techniques to software partitioning, recovery and restructuring*, J. Syst. Softw., 73 (2004), pp. 227–244.

[22] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner, *Bunch: A clustering tool for the recovery and maintenance of software system structures*, in ICSM '99: Proceedings of the IEEE International Conference on Software Maintenance, Washington, DC, USA, 1999, IEEE Computer Society, p. 50.

[23] M. Meila, *Comparing clusterings*, Tech. Report 418, UW Statistics, 2002.

[24] M. Meila, *Comparing clusterings by the variation of information.*, in COLT, B. Schölkopf and M. K. Warmuth, eds., vol. 2777 of Lecture Notes in Computer Science, Springer, 2003, pp. 173–187.

[25] ——, *Comparing clusterings: an axiomatic view.*, in ICML '05: Proceedings of the 22nd International Conference on Machine Learning, 2005, pp. 577–584.

[26] B. Mirkin, *Mathematical Classification and Clustering*, Kluwer Academic Press: Boston-Dordrecht, 1996.

[27] B. Mitchell, *Clustering software systems to identify subsystem structures.* `http://www.mcs.drexel.edu/~bmitchel/research/survey.pdf`, 2001.

[28] J. Neville, M. Adler, and D. Jensen, *Clustering relational data using attribute and link information*, in Proc. of the Text Mining and Link Analysis Workshop, Eighteenth International Joint Conference on Artificial Intelligence, 2003.

[29] J. Neville, M. Adler, and D. Jensen, *Spectral clustering with links and attributes*, Tech. Report UM-CS-2004-042, UMass, 2004.

[30] M. E. J. Newman and M. Girvan, % em Finding and evaluating community structure in networks, Physical Review E, 69 (2004), p. 026113.

[31] A. Sanfeliu, R. Alquézar, and F. Serratosa, *Clustering of attributed graphs and unsupervised synthesis of function-described graphs.*, in ICPR, 2000, pp. 6022–6025.

[32] K. Sartipi and K. Kontogiannis, *A user-assisted approach to component clustering*, Journal of Software Maintenance, 15 (2003), pp. 265–295.

[33] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 888–905.

[34] B. Stein and O. Niggemann, *On the nature of structure and its identification*, in WG '99: Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science, London, UK, 1999, Springer-Verlag, pp. 122–134.

[35] C. Stichting, M. Centrum, and S. V. Dongen, *Performance criteria for graph clustering and markov cluster experiments*, Tech. Report INS-R0012, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, The Netherlands, 2000.

[36] V. Tzerpos and R. C. Holt, *Mojo: A distance metric for software clusterings*, in WCRE '99: Proceedings of the Sixth Working Conference on Reverse Engineering, Washington, DC, USA, 1999, IEEE Computer Society, p. 187.

[37] S. M. v. van Dongen, *Graph Clustering by Flow Simulation*, PhD thesis, University of Utrecht, May 2000.

[38] S. White and P. Smyth, *A spectral clustering approach to finding communities in graph.*, in 2005 SIAM International Conference on Data Mining, 2005.

[39] J. Wu, A. E. Hassan, and R. C. Holt, *Comparison of clustering algorithms in the context of software evolution.*, in ICSM, 2005, pp. 525–535.