GRANULAR GRAPH CLUSTERING IN THE WEB

VASSILIS G. KABURLASOS, LEFTERIS MOUSSIADES,
Dept. of Industrial Informatics, Technological Educational Institution of Kavala
GR-65404 Kavala, Greece

AND ATHENA VAKALI
Dept. of Informatics, Aristotle University of Thessaloniki
GR-54124 Thessaloniki, Greece

We investigate the partition of a weighted graph, representing traffic, to a number of subgraphs such that both inter(external)-subgraph traffic is minimized and intra(internal)-subgraph traffic is maximized. The long-term objective is Web-navigation support. We pursue a solution by applying a simple agglomerative clustering algorithm, or ACA for short, to a metric space emerging from a weighted graph. An enabling technology is inspired from mathematical lattice theory. The proposed techniques compare favorably with other techniques in an application to a graph stemming from a University Web-site.

1. Introduction

In this work we investigate the partition of a weighted graph, representing traffic in the Web, to subgraphs such that "much" of the traffic remains internal within a subgraph. In other words, there is "little" external traffic between subgraphs in the partition. The long-term objective is to use the produced subgraph partition for Web-navigation support.

A number of data management tasks in the Web employ successfully "clustering" techniques in order to facilitate Web user access [16],[18]. There exists an abundance of graph clustering algorithms in various application domains including pattern recognition [5],[9], structure comparison [10],[11], multivariate data processing [1],[8]. Note that the literature is dominated by *divisive* type clustering [2],[6],[19] where clusters are computed "top-down" by successively splitting a master graph. A different type of clustering, namely *agglomerative* clustering, proceeds "bottom-up" by incrementally augmenting graphs. However, agglomerative clustering is not usually pursued in practice mainly due to a shortage of enabling mathematical tools.

Inspired from mathematical lattice theory, this work considers a straightforward agglomerative clustering algorithm (ACA) extension to a metric

space. The latter (space) emerges here from a weighted graph representing traffic in a University Web-site. The proposed techniques compare favourably with alternative clustering techniques from the literature.

The layout is as follows. Section 2 summarizes the mathematical background. Section 3 presents ACA clustering algorithm. Section 4 formulates a real-world problem regarding clustering of Web pages. Section 5 describes data preprocessing as well as experimental results, comparatively. Section 6 summarizes the contribution of this work including also a discussion.

2. Mathematical Background

This section summarizes tools. Mathematical proofs will be detailed elsewhere.

Consider a metric space (S,d). Given $x_i,x_f \in S$, a *shortest path* from x_i to x_f is a sequence $x_i,x_1,\ldots,x_n,x_f$ of "neighboring" set S elements from x_i to x_f such that $d(x_i,x_1)+\ldots+d(x_n,x_f) = d(x_i,x_f)$.

Definition 1: A set $X \subseteq S$ in a metric space (S,d) is called *convex* if and only if $x_i,x_f \in X$ implies that a shortest path from x_i to x_f is also included in X.

Proposition 2. Let $\mathcal{C}$ be the set of convex sets in metric space (S,d). Then $(\mathcal{C},\leq)$ is a mathematical lattice, where $\leq$ denotes set inclusion.

Our interest here focuses on a metric space (S,d) of finite cardinality $|S|$. Useful tools were transferred from lattice theory as shown next.

Let the *unary* operation "$\vee S$" return the supremum of a set S of real numbers; likewise, let the *unary* operation "$\wedge S$" return the infimum of set S. Note that $\vee S$ and $\wedge S$ are simplifications for $\underset{x \in S}{\vee S}$ and $\underset{x \in S}{\wedge S}$, respectively. Obviously, for finite $|S|$, $\wedge S$ and $\vee S$ equal, respectively, the *min* and *max* of set S.

Let d_V: $V \times V \rightarrow \mathbb{R}_0^+$ be a metric. Furthermore, let P_V be the power-set of the (finite) set V. The following four propositions present four metrics in P_V.

Proposition 3: Function $d_a(V_1,V_2)=\frac{1}{|V_1||V_2|}\sum_{i,j} d_V(v_i,v_j)$ is a metric.

Proposition 4: Function $d_M(V_1,V_2)= \underset{i}{\vee}\underset{j}{\vee} d_V(v_i,v_j)$ is a metric.

Proposition 5: Function d_H: $P_V \times P_V \rightarrow \mathbb{R}_0^+$ given by $d_H(V_1,V_2)= \max\{\underset{i}{\vee}\underset{j}{\wedge} d_V(v_i,v_j), \underset{j}{\vee}\underset{i}{\wedge} d_V(v_j,v_i)\}$ is a metric.

Proposition 6: Function d_K: $P_V \times P_V \rightarrow \mathbb{R}_0^+$ given by $d_K(V_1,V_2)= 0.5*[\underset{i}{\vee}\underset{j}{\wedge} d_V(v_i,v_j) + \underset{j}{\vee}\underset{i}{\wedge} d_V(v_j,v_i)]$ is a metric.

The reason for the multiplicative coefficient "0.5" in proposition 6 is in order to produce the "intuitive" equality $d_K(\{a\},\{b\}) = d_V(a,b)$.

The metrics above are different than other ones between graphs [10],[11] in that the latter quantify structural dissimilarity between graphs, whereas the metrics here quantify the distance between graphs. Metrics $d_H(.,.)$ and $d_K(.,.)$ above are extensions of the "Hausdorf" and the "Kaburlasos" metric, respectively, between intervals of real numbers [12],[13],[14]. A metric can be used for implying a *similarity measure* function.

Definition 7: A *similarity measure* is a function $\mu: S\times S\rightarrow(0,1]$ that satisfies

(S1) $\mu(a,b) = 1 \Leftrightarrow a = b$.

(S2) $\mu(a,b) = \mu(b,a)$.

(S3) $\frac{1}{\mu(a,b)}+\frac{1}{\mu(x,x)}\leq\frac{1}{\mu(a,x)}+\frac{1}{\mu(x,b)}$.

A *similarity space* (S,μ) includes both a set S and a similarity measure $\mu: S\times S\rightarrow(0,1]$. A similarity measure can be implied from a metric as follows.

Proposition 8: A metric function $d: S\times S\rightarrow \mathsf{R}_0^+$ implies a similarity measure function $\mu_d: S\times S\rightarrow(0,1]$ given by $\mu_d(a,b) = 1/(1+d(a,b))$.

3. A Simple Clustering Algorithm

The Agglomerative Clustering Algorithm, or ACA for short, shown next, is inspired from a general agglomerative clustering algorithm [3].

Algorithm 1 Agglomerative Clustering Algorithm (ACA)

1: Consider a similarity space (S,μ).
2: Consider a well-defined terminating condition *TC*.
3: Consider n clusters $\mathfrak{G}_1,\ldots,\mathfrak{G}_n$, that is one cluster per metric space element.
4: **while** condition *TC* is not satisfied **do**
   a. Most similar clusters $\mathfrak{G}_I$ and $\mathfrak{G}_J$: $\mu\left(\mathfrak{G}_I,\mathfrak{G}_J\right)=\max_{i,j\in\{1,..,n\},i\neq j}\mu\left(\mathfrak{G}_i,\mathfrak{G}_j\right)$;
   b. Merge $\mathfrak{G}_I$ and $\mathfrak{G}_J$ into a single cluster $\mathfrak{G}_I\vee\mathfrak{G}_J$;
   c. $n\leftarrow n$-1.
5: **end while**
6: Return Clusters.

ACA is a hierarchical algorithm with complexity is $O(n^3)$, where "n" is the cardinality of the training data set. Terminating condition "*TC*" may include a user-defined number of clusters, a user-defined (threshold) similarity measure value μ_0, the disappearance of trivial clusters, etc.

A cluster is interpreted here as an (information) granule [15]. This work considers two types of granules, namely *connected*- and *convex- subgraphs*.

4. A Real-World Problem

Consider a Web-site including V Web-pages with hyperlinks among them. Let a number of users browse the pages. By "traffic" we mean the total number of user traversals from a Web-page to another one in V.

Consider a set $S_0 \subseteq V$ of Web-pages. By *internal traffic* (regarding S_0) we mean the total traffic among Web-pages in S_0. By *external traffic* (regarding S_0) we mean the total traffic between Web-pages belonging to S_0, on the one hand, and Web-pages belonging to $(V \setminus S_0)$, on the other hand.

4.1. *Motivation and Formulation*

Based on evidence (i.e. measurements), our objective is to partition the set V in subsets, namely *clusters*, characterized by both "high" internal traffic and "low" external traffic. Note that between two different partitions with similar internal/external- traffic, preferable is a "finer" one characterized by a larger number of clusters. A "good", in the aforementioned sense, partition can be valuable for supplying Web-navigation support to users.

A Web-site was represented by a "weighted graph" such that a graph-vertex corresponded to a Web-page, a graph-link corresponded to a hyperlink (from a Web-page to another one); furthermore, the weight of a graph-link was a function of the corresponding number of user traversals.

5. Experiments and Results

We collected data from the Web-site of the Informatics Department of Aristotle University of Thessaloniki, Greece during a period of four months from March to June 2005. A master-graph was synthesized from 3,354,452 user requests recorded in the server's "log file" as described next.

5.1. *Data Preprocessing*

The server's "log file" is a text file, which records every user request [4]. The format of the aforementioned file, typically employed by a HTTP server, is the *common logfile format* (*CLF*). Data preprocessing was carried out in three steps including (1) Data cleaning, (2) Organization of requests into user sessions, and (3) Identification of site structure.

In conclusion, a weighted master-graph was produced including 785 vertices and 4,582 links in 17 disconnected subgraphs. We defined the weight $w_{i,j}$ of a link from vertex "i" to vertex "j" as the inverse of the number of all user

traversals from "*i*" to "*j*". In other words, the number of all user traversals from vertex "*i*" to vertex "*j*" equals $1/w_{i,j}$.

Definition 9: The *inner-transactions ratio* (*ITR*) index of a master-graph partition is the ratio of the sum of weights (of links having both their endpoints) in the same cluster over the sum of all link weights in the master-graph.

We remark that ITR is a straightforward extension for weighted graphs of the *Coverage index* [2]. Like the Coverage index, the ITR should be used for the evaluation of a clustering together with additional indices including the number/size of clusters.

5.2. *Experiments*

We carried out a large number of experiments, comparatively.

We used Floyd's algorithm [7] to compute the distance between two vertices in a graph. We applied ACA using the four metrics d_a, d_M, d_H, and d_K. For comparison, we applied another (non-metric) distance, that is $d_S(V_1,V_2)= \wedge_i \wedge_j d_V(v_i, v_j)$. We also applied algorithm *RandClust*, which merged clusters at random. The latter algorithm was meant to investigate empirically how much "better than random" our proposed ACA algorithms may perform.

Figure 1 plots the corresponding ITR values as agglomeration proceeds, i.e. the no. of clusters reduces. Apparently, the proposed ACA algorithms clearly provide high ITR values (near 1) for fewer than 500 no. of clusters. In particular, metrics d_H and d_K have demonstrated a marginal advantage.

Random clustering performed poorly. More specifically, high ITR values were recorded for algorithm *RandClust* only for less than 100 no. of clusters (Figure 1). For the reader's interest Figure 2 shows how cluster population statistics (including mean and standard deviation) change for an ACA algorithm as the agglomeration proceeds. Figure 2 also shows that a no. of clusters between 200 and 100 corresponds to both "desirably high ITR values" (near one) and a "desirable cluster size" (around 40). The latter size is considered appropriate for Web navigation user support in the context of this work.

Table 1 displays statistics for various clustering methods for both 200 and 100 no. of clusters. For comparison we applied an extension of the divisive *MajorClust* algorithm [19] for graph clustering, namely *WMajorClust* [17].

Table 1 confirms the poor performance of random clustering (*RandClust*) characterized by small ITR values near zero. Table 1 also demonstrates that divisive algorithm *WMajorClust* produced an ITR value of 0.90, clearly inferior to the corresponding ITR values near 0.99 supplied by the agglomerative clustering (ACA) algorithms.

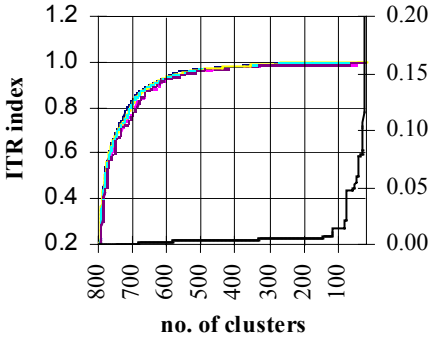

Figure 1. The ITR values of five ACA clustering methods based on d_S, d_H, d_K, d_a, and d_M, respectively, asymptotically approach 1 for fewer than 700 no. of clusters. To the contrary, the ITR values by method *RandClust* (shown by the curve to the lower-right) increase fast to 1 only for fewer than 100 no. of clusters.

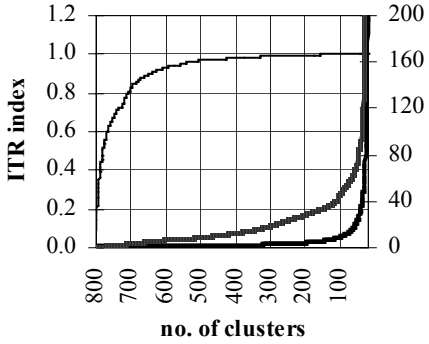

Figure 2. Cluster statistics including *mean* (dark line: lower-right down) and *standard deviation* (light line: lower-right up) are shown for ACA by d_a. The corresponding ITR curve is also shown to the upper-left. Similar curves were also obtained by the other ACA clustering methods used here.

The previous experiments computed, in general, non-convex subgraph granules. We repeated the experiments again, this time computing only convex subgraph granules. In a series of experiments we confirmed that the total number of computed subgraphs reduced consistently by 5% to 10%. Moreover, the ITR index consistently improved (i.e. increased) in the second- or the third-digit. Hence, we conclude that convex clustering improved performance here.

6. Discussion and Conclusion

We introduced four different metrics, i.e. d_a, d_M, d_H, d_K, which implied four similarity measures, between sets of points in a metric space. Those similarity measures were employed by an agglomerative clustering algorithm (ACA) for computing a subgraph partition in a weighted master-graph.

We demonstrated an application to a metric space stemming from a weighted master-graph representing traffic in a University Web-site. We achieved a good partition to subgraphs such that both the inter(external)-subgraph traffic was minimized and the intra(internal)-subgraph traffic was maximized. Our algorithms could be used for fast computing an approximate solution to the "minimum cut" problem [6] as it will be shown in a future work.

Table 1. Indices of performance for various clustering methods (M).

M	no. of clusters	Cluster statistics min	max	mean	std	ITP
d_a	200	1	366	3.92	26.15	0.99
	100	1	380	7.85	38.53	0.99
d_H	200	1	313	3.92	22.66	0.98
	100	1	313	7.85	32.51	0.99
d_M	200	1	274	3.92	20.42	0.99
	100	1	274	7.85	29.38	0.99
d_K	200	1	345	3.92	24.71	0.99
	100	1	345	7.85	35.48	0.99
d_S	200	1	451	3.92	32.00	0.99
	100	1	451	7.85	45.70	0.99
RandClust	200	1	42	3.92	5.58	0.00
	100	1	96	7.85	12.66	0.01
WMajorClust (statistics in 100 runs)	55.94	2	215.	14.0	30.54	0.90

References

1. F.B. Baker and L.J. Hubert, *J. Amer. Statist. Assoc.* **71**, 870 (1976).
2. U. Brandes, M. Gaertler, D. Wagner, *LNCS* **2832**, 568. Springer (2003).
3. R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, 2nd ed. New York, NY: Wiley & Sons (2001).
4. F.M. Facca and P.L. Lanzi, *Data & Knowl. Eng.* **53**, 225 (2005).
5. M. Fernandez and G. Valiente, *Pat. Recog. Let.* **22**, 753 (2001).
6. G.W. Flake, R.E. Tarjan, K. Tsioutsiouliklis, *Internet Math.* **1**, 385 (2004).
7. R.W. Floyd, *Communications* **5**, 345 (1962).
8. J.C. Gower and G.J. Ross, *Appl. Stat.* **18**, 54 (1969).
9. S. Gunter and H. Bunke, *Pat. Recog. Let.* **23**, 405 (2002).
10. S. Gunter and H. Bunke, *Pat. Recog. Let.* **24**, 1107 (2003).
11. D. Justice, A. Hero, *IEEE Trans. Pat. Anal. Mach. Intel.* **28**, 1200 (2006).
12. V.G. Kaburlasos, Studies in Computational Intelligence **27**. Heidelberg, Springer (2006).
13. V.G. Kaburlasos and A. Kehagias, *IEEE Trans. Fuzzy Syst.* **15**, 243 (2007).
14. V.G. Kaburlasos and S.E. Papadakis, *Neural Netw.* **19**, 623 (2006).
15. T.Y. Lin, *Intl. J. Approx. Reas.* **40**, 1 (2005).
16. M. Perkowitz and O. Etzioni, *Artif. Intel.* **118**, 245 (2000).
17. L. Moussiades and A. Vakali, *The Computer Journal* **48**, 651 (2005).
18. K.A. Smith and A. Ng, *Dec. Sup. Syst.* **35**, 245 (2003).
19. B. Stein and O. Niggemann, *LNCS* **1665**, 122. Heidelberg, Springer (1999).