Semi-supervised Graph Rewiring with the Dirichlet Principle

Manuel Curado, Francisco Escolano and Miguel A. Lozano
Department of Computer Science and AI
University of Alicante
Alicante, Spain
Email: {mcurado, escolano, malozano}@dccia.ua.es

Edwin R. Hancock
Department of Computer Science
University of York
York, UK
Email: edwin.hancock@york.ac.uk

***Abstract*—In this paper, we propose the concept of *graph rewiring* and we show how to exploit it in an un-supervised setting so that commute times can be better estimated by state-of-the-art methods. Our experiments show a significant improvement with respect to unsupervised graph rewiring.**

I. Introduction

In this paper, we coin the term *graph rewiring* to denote several techniques (e.g. densification, sparsification, compression-decompression) addressed to process an input graph so that the subsequent pattern recognition task is better conditioned. Given $G = (V, E)$, graph densification [1] aims at generating $H = (V, E')$ where $E' \supseteq E$ and the cuts in G are preserved (or bounded) to some extent in H . This is interesting in graph-based manifold learning where the input graphs (typically k-NN or Gaussian) are very sparse. Denser graphs with more intra-class links and a minimal overhead of inter-class links are thus better conditioned for clustering and ranking.

On the other hand, graph sparsification [2] works in the opposite sense: we seek $E' \subseteq E$ in H, and in particular $|E'| \ll |E|$. This is key in shape simplification, where we want to have a principled skeleton of an input graph to boost the efficiency of graph matching or the computation of graph similarities. Finally, compression-decompression via the Regularity Lemma [3], [4] is a promising tool for facing big-data from a graph-based perspective (see recent clustering experiments in [5]).

A good benchmark to test graph rewiring is the measurement of vertex similarity. It is well known that commute times (CTs) are reduced to meaningless local measures in medium-size/large graphs [6]. It has been observed [7] that this is partially due to the fact that CT_{ij} for vertices i and j is the solution to an optimization problem involving a $p = 2$ norm. Namely, the resistance distance $R_{ij} = \frac{1}{2|E|} CT_{ij}$ comes from $R_{ij} \triangleq \arg\min_Y \sum_{e \in E} r_e |y_e|^p$, where $p = 2$ and $Y = \{y_e\}_{e \in E}$ is the unit flow from i to j (inject a unit current at i, extract it at j and observe the flow traced across the edges $e \in E$). Unit flows have two interesting properties. First, they are quite scattered along the edges (even in moderate size graphs), and second, the bulk of their magnitude is in the neighborhood of both i and j. This is why Nguyen and Mamitsuka [8] proposed to relax p to the unit. With $p < 2$ they amplify the global (scattered part) of the flow making it more comparable to the local part (flow in the neighborhoods of i and j). Best results are obtained with $p = 1$. However, this maybe not enough if the graph is not processed beforehand.

The need of pre-processing the graph is better understood when one studies the interplay between sparsification and densification. Some state-of-the-art sparsifiers are edge samplers where the weight/importance of each edge is measured in terms of the resistance distance[9], [2]. However, how can one obtain a good sparsification if the resistance measure itself is prone to both flow scattering and local flow concentration?

Our preliminary results with alpha shapes [10] show that the construction of the representation may involve well structured wrong links between parts of the object and the whole. In this context, alleviating scattering (amplifying the global part of the flow) is not enough. We must confine undesirable diffusions (say *inter-class edge noise*) to the extent of the object's parts. This is what graph densification does: it produces/predict edges mostly within clusters at the cost of some inter-class diffusion. In addition, graph densification is closely related with the Regularity Lemma, since the strong version of the lemma only works for dense graphs. Our recent experiments show that finding regular partitions and then reconstructing the graph can be seen as an inter-class noise filter [11]. However, the input graph has to be dense and/or densified beforehand.

Graph densification was originally posed in terms of Semi-definite programming (SDP) [1]. Since densification involves $O(n^2)$ variables, where $n = |V|$, and SDP solvers are polynomial in the number of unknowns, the problem is intractable for medium-size/large graphs (the range where CTs are meaningless). Our experiments [12] also show that the energy function is too weak to constrain undesirable (inter-class) diffusions even for small graphs. This motivated the development of an unsupervised and scalable method for graph densification. Dirichlet graph densifiers [13] rely on two elements: a) return random walks and b) a Dirichlet process. Return random walks are designed to retain the diffusion process inside each cluster (i.e. with minimal intra-class linkage). The Dirichlet process drives, in the worst case, on the $O(n^4)$ Laplacian associated with the edges. However, in practice we can obtain good densifications (from the point of view of CTs estimation) with Laplacians of order $O(\alpha n^4)$ with $\alpha \approx 0.35$.

II. Contributions

In this paper we show that semi-supervised densification leads to a) boosting the quality of graph densification, and b) reducing α significantly. In addition, we show that graph rewiring is beyond the concept of graph densification: good estimations of CTs can be achieved with actually less dense graphs. Herein, we show how the Dirichlet principle contributes to an *intelligent diffusion*. Intelligent diffusion is illustrated by the experimental evidence showed in the paper.

III. Related work

This work is closely related to the concept of *anchor graphs* [14][15]. Anchor graphs provide a *data-to-anchor* kNN structure governed by a set of $m \ll n = |V|$ representatives (anchors) typically obtained through K-means clustering, in $O(dmnT + dmn)$ time where $O(dmnT)$ is due to the T iterations of the K-means process. These graphs tend to make out-of-the-sample predictions compatible with those of Nyström approximations, and in turn their approximated adjacency/affinity matrices are ensured to be positive semidefinite.

In [19], the predictive power of non-parametric regression rooted in the anchors/landmarks ensures a way of constructing very informative weighted kNN graphs. Since anchor graphs are bipartite (only *data-to-anchor* edges exist), this representation bridges the sparsity of the pattern space because a random walk traveling from node u to node v must reach one or more anchors in advance. In other words, for a sufficient number of anchors it is then possible to find links between distant regions of the space. This opens a new perspective for computing meaningful commute distances in large graphs. It is straightforward to check that the spectral properties of the approximate weight matrix $W = Z\Lambda Z^T$, where $\Lambda = \mathrm{diag}(Z^T 1)$ and Z is the data-to-anchor mapping matrix, rely on its low-rank. Then, it is possible to compute a reduced number of eigenvalue-eigenvector pairs associated with $M = \Lambda^{-1/2} Z^T Z \Lambda^{-1/2}$, which has also dimension $m \times m$ and where m is the number of anchors. This leads to a compact solution for the spectral hashing problem [20] (see [21] for details). In this way, these eigenvectors-eigenvalues may also provide a meaningful estimation of the commute distances between the samples through the spectral expression of this distance [22]. Our interpretation is that the goodness of the eigenvalue-eigenvector pairs is a consequence of performing kernel PCA process over ZZ^T where the columns of Z act as kernel functions. This interpretation is consistent with the good hashing results obtained with anchor graphs [23][21] where the kernel encoded in the columns of Z is extensively exploited.

Thus, anchor graphs are a good representation, a particular case of graph densification/graph rewiring. They are appealing but one needs to find the optimal number and placement of the anchors. This motivated us to look for a more flexible representation for conditioning the graphs by rewiring their edges properly.

IV. Semi-Supervised Dirichlet Graph Rewiring

Semi-Supervised Graph Rewiring consists of determining what edges must be asked to the supervisor so that the quality of the graph is improved. Dirichlet processes ensure that the resulting graph is more suitable for measuring CTs than the original graph.

Our approach consists of the following steps:

1) **Knn-graph**: Given a data set $\chi = \{\vec{x}_1, ..., \vec{x}_n\} \subset \mathbb{R}^d$, we map the $\vec{x}_i$ to the vertices V of an undirected weighted graph $G(V, E, W)$ with $W_{ij} = e^{-||\vec{x}_i - \vec{x}_j||^2/\sigma^2}$ and $(i, j) \in E$ if $W_{ij} > 0$ and $j \in N_k(i)$.
2) **Labelled Links:** Select a set of potential links $\mathcal{L} = \{(a, b) \in V \times V, (a, b) \notin E\}$ and set $W_{ab} = 1$ if $(a, b) \in \mathcal{L}$. Thus, we have $E' = E \cup \mathcal{L}$. The set $\mathcal{L}$ is chosen according to: a) priorizing classes c whose edge density $e(c)$ satisfies $e(c) + \beta_c = \bar{e}(G)$, where $\bar{e}(G)$ is the average edge density, and b) setting $|\mathcal{L}| = \beta_c |c|$.
3) **Return Random Walk:** Given $G' = (V, E', W)$ reformulate W in terms of W' so that

$$W'_{ij} = \max_k \max_{\forall l \neq k} \{p_{v_k}(v_j|v_i) p_{v_l}(v_i|v_j)\} , \qquad (1)$$

where $p_{v_k}(v_j|v_i) = \frac{W_{ik} W_{kj}}{d(v_i) d(v_j)}$, $p_{v_l}(v_i|v_j) = \frac{W_{jl} W_{li}}{d(v_j) d(v_i)}$ (*go* and *return* probabilities, respectively) and $d(.)$ is the degree function. Therefore, W'_{ij} relies on maximizing the probability that a random walk goes from i to j through l and then returns through a different vertex k. This strategy minimizes the weight of spurious inter-class links.
4) **Edge Selection:** Given $G' = (V, E', W')$, select $E'' \subset E$, with $|E"| \ll |E|$ as follows:
   a) $\mathcal{S} = \mathbf{sort}(E, W'_e, descend)$.
   a) $\mathcal{S}' = \mathcal{S} \sim \{e \in \mathcal{S} : W'_e < \delta_1\}$ where δ_1 is set so that $|\mathcal{S}'| = \alpha |\mathcal{S}|$.
5) **Line Graph** Given $G'' = (V, \mathcal{S}', W')$ construct a the graph $Line = (\mathcal{S}', Line_E, Line_W)$ where
   a) The nodes of $e_i \in Line$ are the edges in $\mathcal{S}'$.
   b) The weight function $Line_W$ is defined as follows:

$$Line_W(e_a, e_b) = \sum_{k=1}^{|E"|} p_{e_k}(e_b|e_a) p_{e_k}(e_a|e_b) , \qquad (2)$$

      i.e. we use *go* and *return* probabilities.
   c) $Line_E = \{(e_a, e_b) : Line_W(e_a, e_b) > 0\}$
6) **Dirichlet Process:** Given the $Line$ graph, we proceed as follows:
   a) $\mathcal{SB} = \mathbf{sort}(\mathcal{S}', Line_W, descend)$.
   b) $\mathcal{SB}' = \mathcal{SB} \sim \{e \in Line_E : Line_W < \delta_2\}$ where δ_2 is set so that $|\mathcal{SB}'| = \beta |\mathcal{SB}|$.
   c) Consider $\mathcal{SB}'$ as the boundary B (known labels) of a Dirichlet process driven by the Laplacian $Line_{\mathcal{L}} = Line_D - Line_W$. Then, finding an

harmonic function, i.e. a function $u(.)$ satisfying $\nabla^2 u = 0$ consists of minimizing:

$$D_{Line}[u] = \frac{1}{2} u^T Line_{\mathcal{L}} u \tag{3}$$

where $u = [u_B, u_I]$ and $Line_{\mathcal{L}}$ are re-ordered so that the boundary nodes (edges in $Line$) come first. Then, minimizing $D_{Line}[u]$ wrt u_I leads to label of the unknown nodes (edges in $Line$) u_I as the solutions to the following linear system:

$$L_I u_I = -K^T u_B \ , \tag{4}$$

where all the u_B are set to the unit, L_I is the sub-Laplacian of $Line_{\mathcal{L}}$ concerning the u_I nodes, and K is a $|\mathcal{SB}'| \times |\mathcal{SB}'|$ block of the re-ordered Laplacian.

7) **Relabelling:** Since there is a bijection between the nodes in the line graph and the edges in the original graph, we relabel the edges in the orginal graph with the information coming from the Dirichlet process in the line graph.

V. Experiments

Our goal is to increase the number in edges in classes whose density is lower than the global inter-class density. In [13], we presented an unsupervised graph densification method that notably improves the results obtained with kNN. This method yields better results selecting a high number of edges in the Laplacian matrix (35%), thus obtaining a densified matrix with a higher number of edges than the input kNN. However, these results may be improved by introducing a previous supervised step to densify the graph, yielding an smaller and less dense Laplacian. Next, we show the obtained results.

The strategy followed to add intra-class edges is as follows: (i) The density of each class is obtained individually and it is compared to the global average intra-class density of all the classes in the graph. (ii) Edges are added to classes whose individual density is lower than the global one, until they reach the global density. These edges are added randomly and their value is 1. (iii) Our Dirichlet Densification method is applied to densify/rewire the resulting graph. Finally, we run the robust method for estimating CTs in the rewired graph. Performance is measured in terms of Adjusted Rand Index (ARI) of the affinity matrix associated with the CTs.

Next, we will present a set of experiments consisting of applying this process to different datasets.

A. Experiment 1: Partial local density: increasing the density of a single class

The goal of the first experiment is to prove that a previous densification of the class with the lowest density may improve the results. We'll use the NIST dataset, using as a reference the best kNN obtained in our preliminary experiments ($k = 15$). The obtained results are shown in Figure 1. The top-left element in the figure displays the densification level (percentage) for each of the 10 dataset classes, and the dotted horizontal red line represents the global average density for all the classes. We can see that class #3 has approximately 3% less density than the average. We can also see the corresponding kNN matrix (top-center), and the best densification obtained with our unsupervised method (top-left), corresponding to use a 35% of the Laplacian matrix, and a 5% of known edges. In the bottom row we show the results obtained by adding a 3% of 1-valued edges to class #3 in a previous step, as we can see in the kNN matrix (bottom-center). Finally, we apply our method to densify the graph (bottom-right).

In our preliminary unsupervised experiments, an increase in the number of edges yields a final densified matrix with a higher number of edges than the original one. However, adding edges locally in a particular class yields a different result: the obtained matrix is less dense than the input matrix (previous to densification). This is due to the fact that, in principle, the 1-valued edges added in RRW cause that most of the paths generated within the class go through these edges, thus reducing the probability of choosing paths that do not include them. In further steps of the rewiring, when our method chooses the edges with higher weights in the line graph, it includes the intra-class edges of this class. This reduces the inter-class noise and the confusion with other classes. Concluding, we obtain a better result with a lower number of edges than the input graph, due to the smart label selection and the intelligent Dirichlet diffusion.

B. Experiment 2: Increasing global intraclass density

The second experiment consists of studying the effect of the densification of all the classes with a density lower than the average one. We use the same dataset than in the previous experiment, but in this case we increase the density of classes #3, #4, #5, #6, #8, #9, and #10, by increasing their number of edges in 2.61%, 0.64%, 0.8%, 2.52%, 0.19%, 0.42% and 1.15% respectively. Thus, there are not classes below the average density, as we can see in Figure 2 (bottom-left). We also show in this figure the input graph obtained with this densification (bottom-center), and the densified graph (bottom-right), with an adjusted rand index of 82.91% with $E'' = 0.3$. We have obtained an improvement close to 10%.

C. Experiment 3: Analysis of the densification level

In this experiment, we study which is the necessary densification level to improve the best result provided by the unsupervised method, and if is it possible to use a smaller Laplacian matrix so that we outperform the unsupervised method for different datasets.

We define P as a factor that determines how the difference of density percentage with respect to the average will be reduced. E.g., defining $P = 0.5$ (50%) in a class with an initial density of 4% below the average, means that we will add edges until the density if 2% below the average.

Next, we analyze the results of the datasets (NIST, COIL and LOGOS) for different values of P (0.25, 0.33, 0.5 and 1).

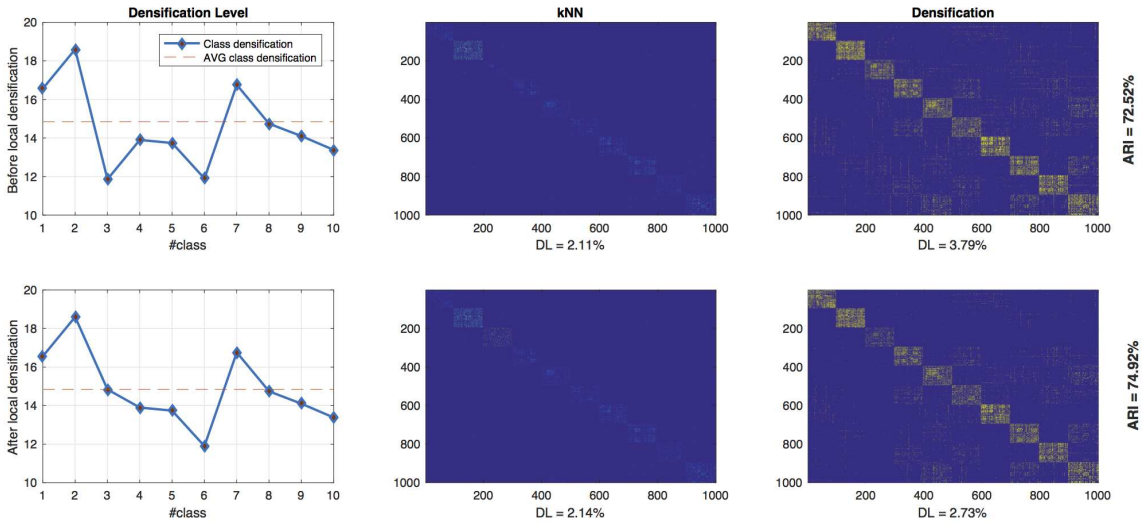

Fig. 1. Nist dataset: Increasing local density of class three with a size of laplacian equal to 0.35. The best result of NIST unsupervised is 72.52% and semisupervised result is 74.92%

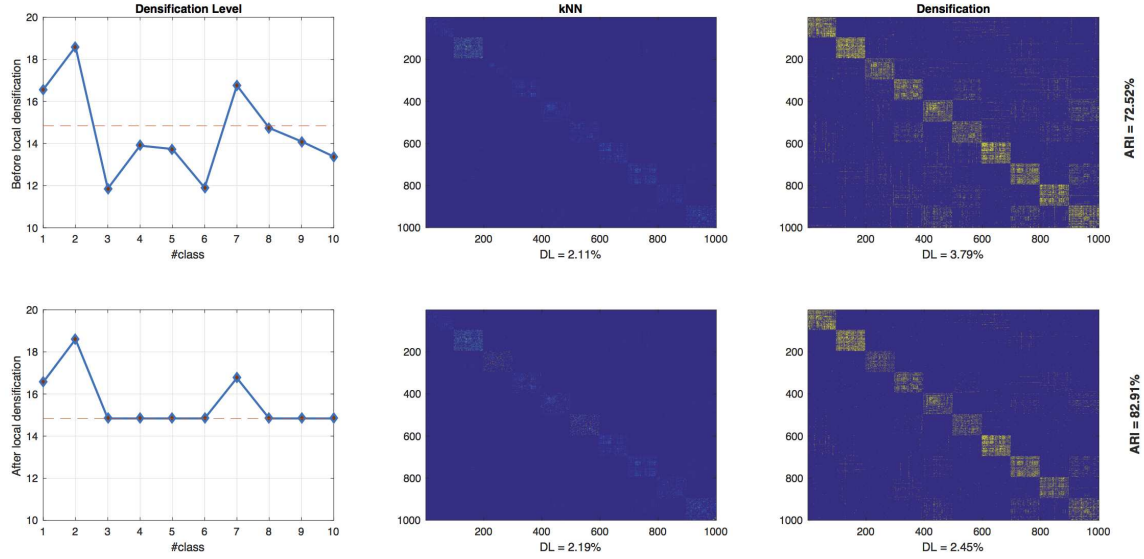

Fig. 2. Nist dataset: Increasing local density of all classes with a size of laplacian equal to 0.3. The best result of NIST unsupervised is 72.52% and semisupervised result is 82.91%

1) NIST: This dataset consist of 10 classes with 100 elements per class, with an average density of 14.7%, and a median density of 14%. Regarding the local density, 7 of these 10 classes are below the average density.

We analyze different densifications for the indicated P values and different Laplacian sizes (0.05, 0.1, 0.15, 0.2, 0.25, 0.3 and 0.35), and we obtain the Adjusted Rand Index for each case. The results are shown in table I, highlighting in bold all the cases improving the best result yielded by the unsupervised method (72.52%). In Figure 2 (bottom-right) we show the best case, with and Adjusted Rand Index of 82.91% and a densification level of 2.45%, notably lower than the best result of the unsupervised method (3.79%).

These results show that we may obtain better rewirings with a smaller Laplacian (see Figure 3). We can see that as we increase P, and thus the number of edges of the input matrix, we obtain a less dense result but with lower inter-class noise. Therefore, the denser the input classes, the more efficient is the assignment of edges in the densification.

TABLE I
NIST DATASET: ADJUSTED RAND INDEX FOR DIFFERENT PERCENTAGE OF ADDED EDGES AND SIZE OF LAPLACIAN MATRICES

	0.05	**0.1**	**0.15**	**0.2**	**0.25**	**0.3**	**0.35**
P=0.25	71.59	65.9	69.79	71.04	71.98	**73.12**	**73.53**
P=0.33	60.55	60.98	69.98	**73.69**	**74.46**	**73.65**	**73.16**
P=0.5	60.44	70.68	71.33	**74.38**	**74.33**	**74.74**	71.8
P=1	71.59	**76.88**	**78.35**	**79.96**	**79.19**	**82.91**	**82.38**

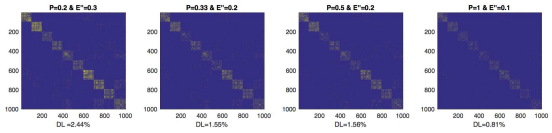

Fig. 3. NIST dataset: four cases with Laplacian size lower than the unsupervised case, that also improve their results (72.56%). The results of Adjusted RI are (left-to-right) 73.12%, 73.69%, 74.38% and 76.88% (in red in table I)

TABLE II
COIL-20 DATASET: ADJUSTED RAND INDEX FOR DIFFERENT PERCENTAGE OF ADDED EDGES AND SIZE OF LAPLACIAN MATRICES

	0.05	**0.1**	**0.15**	**0.2**	**0.25**	**0.3**	**0.35**
P=0.25	69.47	71.02	72.08	76.59	77.01	76.89	69.18
P=0.33	69.53	77.27	78.28	78.97	77.48	**96.44**	**97.75**
P=0.5	76.66	74.64	75.09	80.1	75.66	75.13	**98.12**
P=1	75.09	75.01	69.22	75.83	74.2	**99**	**99.41**

2) COIL: The COIL-20 dataset[1] consists of 20 classes with 72 elements per class [24], with an average density of 21.33%, and a median density of 23.01%. Regarding the local density, 7 of these 20 classes are below the average density.

This dataset has less inter-class noise than the previous one, and therefore the obtained improvement with the semisupervised method is lower, because the initial density is appropriate, as we can see in table II. However, in this dataset is more difficult to improve the unsupervised method with smaller Laplacians because of the structure of the graphs: the first classes are denser than the others. This causes a notable imbalance. We can see that the median is quite different from the average. Building a small Laplacian in this scenario implies to obtain a reduced sample of the original set of edges, in which the selected edges will be the ones with higher weights, and these edges mainly correspond to the first classes (with higher densities). This causes that the other classes are not correctly densified.

3) LOGO: The LOGO (FlickrLogos-32[2]) consists of 32 classes with 70 elements per class [25], with an average density of 20.19%, and a median density of 20.53%. Regarding the local density, 19 of these 32 classes are below the average density.

The best result obtained with the unsupervised method for this dataset es 62.96%, with is notably improved with the unsupervised method as we can see in Table III. The best semisupervised result is obtained with $P = 1$ and a Laplacian $L = 35\%$, and its Adjusted Rand Index is 77.79%. In this table we highlight in bold the cases in which the supervised method improves the unsupervised one. In some of them a small Laplacian with $L = 10\%$ allows to reach an improvement of about a 5% (ARI = 67.1%).

[1]http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php
[2]http://www.multimedia-computing.de/flickrlogos/

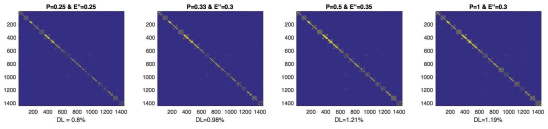

Fig. 4. COIL-20 dataset: four cases with Laplacian size lower then the unsupervised case, most of them improving their results (95.44%). The results of Adjusted RI are (left-to-right) 99%, 98.12%, 96.44% and 77.01%

TABLE III
LOGO DATASET: ADJUSTED RAND INDEX FOR DIFFERENT PERCENTAGE OF ADDED EDGES AND SIZE OF LAPLACIAN MATRICES

	0.05	**0.1**	**0.15**	**0.2**	**0.25**	**0.3**	**0.35**
P=0.25	54.46	61.45	**63.64**	**64.02**	**64.54**	**65.34**	**64.16**
P=0.33	58.11	**63.36**	**66.18**	**66.89**	**65.97**	**66.99**	**66.45**
P=0.5	61.15	**68.12**	**69.95**	**71.2**	**70.03**	**69.83**	**69.82**
P=1	**66.75**	**73.79**	**75.97**	**76.93**	**77.45**	**77.22**	**77.79**

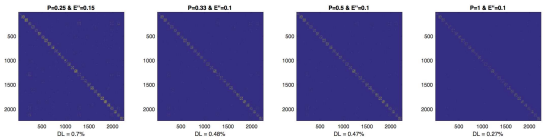

Fig. 5. LOGO dataset: four cases with Laplacian size lower then the unsupervised case, that also improve its results (62.96%). The results of Adjusted RI are (left-to-right) 63.64%, 63.36%, 68.12% and 66.75%

D. Experiment 4: Comparison with Anchor Graphs

As we have stated in the introduction, anchor graphs rely on selecting the number of placement of all the anchors. Their number m has to be determined empirically, whereas the placement is given by K-means clustering. In Figure 6-right we show the evolution of the ARI as we increase the number of anchors, run K-means, and compute CTs on the NIST dataset. The optimal performance of anchor graphs is given for an intermediate number of anchors. However, our unsupervised method clearly outperforms anchor graphs but in a reduced interval. When the supervised method proposed in this paper is applied, we outperform anchor graphs in the whole spectrum.

A key property of rewired graphs is the fact that their spectral gap is minimized with respect to that of anchor graphs (see Figure 6)-left. This is important since local estimations of commute times are bounded by the inverse of the spectral gap. Thus, if the spectral gap is close to zero, then local estimations are unbounded and we are closer to the real commute times.

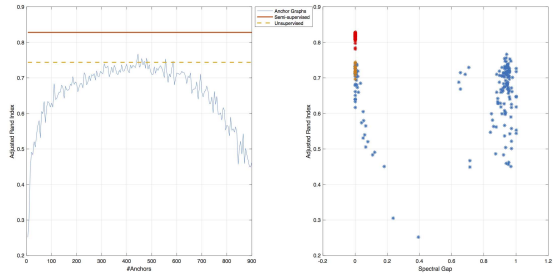

Fig. 6. Comparison with Anchor Graphs (NIST dataset). Left ARI: vs number of anchors. Right: ARI vs spectral gaps.

VI. Conclusion

In this paper, we have contributed with a semi-supervised method for rewiring graphs. In this context, graph rewiring exploits the intelligent labelling driven by the Dirichlet principle to improve the quality of the input graph. Such improvement allows us to measure vertex similarities such as Commute Times in better topological conditions. Our extensive experiments show that rewiring is a more general concept than densification and also that it often leads to reduce the size of the Laplacian associated with the line graph to achieve results comparable to those obtained by the unsupervised version of the method.

Acknowledgements

M. Curado, F. Escolano and M.A. Lozano are funded by the project TIN2015-69077-P of the Spanish Government.

References

[1] M. Hardt, N. Srivastava, and M. Tulsiani, "Graph densification," in *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, 2012, pp. 380–392. [Online]. Available: http://doi.acm.org/10.1145/2090236.2090266

[2] J. D. Batson, D. A. Spielman, N. Srivastava, and S. Teng, "Spectral sparsification of graphs: theory and algorithms," *Commun. ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[3] J. Komlós, A. Shokoufandeh, M. Simonovits, and E. Szemerédi, "The regularity lemma and its applications in graph theory," in *Theoretical Aspects of Computer Science, Advanced Lectures (First Summer School on Theoretical Aspects of Computer Science, Tehran, Iran, July 2000)*, 2000, pp. 84–112.

[4] A. M. Frieze and R. Kannan, "The regularity lemma and approximation schemes for dense problems," in *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, 1996, pp. 12–20.

[5] M. Pelillo, I. Elezi, and M. Fiorucci, "Revealing structure in large graphs: Szemerédi's regularity lemma and its use in pattern recognition," *Pattern Recognition Letters*, vol. 87, pp. 4–11, 2017.

[6] U. von Luxburg, A. Radl, and M. Hein, "Hitting and commute times in large random neighborhood graphs," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1751–1798, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2638591

[7] M. Alamgir and U. von Luxburg, "Phase transition in the family of p-resistances," in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, 2011, pp. 379–387.

[8] C. H. Nguyen and H. Mamitsuka, "New resistance distances with global information on large graphs," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, 2016, pp. 639–647.

[9] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," *SIAM J. Comput.*, vol. 40, no. 6, pp. 1913–1926, 2011. [Online]. Available: http://dx.doi.org/10.1137/080734029

[10] F. Escolano, M. Curado, S. Biasotti, and E. R. Hancock, "Shape simplification through graph sparsification," in *Graph-Based Representations in Pattern Recognition - 11th IAPR-TC-15 International Workshop, GbRPR 2017, Anacapri, Italy, May 16-18, 2017, Proceedings*, 2017, pp. 13–22.

[11] M. Fiorucci, A. Torcinovich, M. Curado, F. Escolano, and M. Pelillo, "On the interplay between strong regularity and graph densification," in *Graph-Based Representations in Pattern Recognition - 11th IAPR-TC-15 International Workshop, GbRPR 2017, Anacapri, Italy, May 16-18, 2017, Proceedings*, 2017, pp. 165–174.

[12] F. Escolano, M. Curado, and E. R. Hancock, "Commute times in dense graphs," in *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2016, Mérida, Mexico, November 29 - December 2, 2016, Proceedings*, 2016, pp. 241–251.

[13] F. Escolano, M. Curado, M. A. Lozano, and E. R. Hancock, "Dirichlet graph densifiers," in *Structural, Syntactic, and Statistical Pattern Recognition - Joint IAPR International Workshop, S+SSPR 2016, Mérida, Mexico, November 29 - December 2, 2016, Proceedings*, 2016, pp. 185–195.

[14] W. Liu, J. Wang, and S. Chang, "Robust and scalable graph-based semisupervised learning," *Proceedings of the IEEE*, vol. 100, no. 9, pp. 2624–2638, 2012. [Online]. Available: http://dx.doi.org/10.1109/JPROC.2012.2197809

[15] W. Liu, J. He, and S. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, 2010, pp. 679–686.

[16] U. von Luxburg and M. Alamgir, "Density estimation from unweighted k-nearest neighbor graphs: a roadmap," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, 2013, pp. 225–233.

[17] M. Alamgir and U. von Luxburg, "Shortest path distance in random k-nearest neighbor graphs," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[18] U. von Luxburg, A. Radl, and M. Hein, "Getting lost in space: Large sample analysis of the resistance distance," in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, 2010, pp. 2622–2630.

[19] D. Cai and X. Chen, "Large scale spectral clustering via landmark-based sparse representation," *IEEE Trans. Cybernetics*, vol. 45, no. 8, pp. 1669–1680, 2015. [Online]. Available: http://dx.doi.org/10.1109/TCYB.2014.2358564

[20] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, 2008, pp. 1753–1760.

[21] W. Liu, J. Wang, S. Kumar, and S. Chang, "Hashing with graphs," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, 2011, pp. 1–8.

[22] H. Qiu and E. R. Hancock, "Clustering and embedding using commute times," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1873–1890, 2007. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2007.1103

[23] W. Liu, C. Mu, S. Kumar, and S. Chang, "Discrete graph hashing," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, 2014, pp. 3419–3427.

[24] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia university image library (coil-20)," 1996. [Online]. Available: http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[25] S. Romberg, L. G. Pueyo, R. Lienhart, and R. van Zwol, "Scalable logo recognition in real-world images," in *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ser. ICMR '11, 2011.