Dirichlet Densifiers for Improved Commute Times Estimation

Manuel Curado[a], Francisco Escolano[a], Miguel A. Lozano[a], Edwin R. Hancock[b]

[a]*Department of Computer Science and AI, University of Alicante, 03690, Alicante (Spain)*
[b]*Department of Computer Science, University of York, York, YO10 5DD, UK*

Abstract

In this paper, we develop a novel *Dirichlet densifier* that can be used to increase the edge density in undirected graphs. Dirichlet densifiers are implicit minimizers of the spectral gap for the Laplacian spectrum of a graph. One consequence of this property is that they can be used improve the estimation of meaningful commute distances for mid-size graphs by means of topological modifications of the original graphs. This results in a better performance in clustering and ranking. To do this, we identify the strongest edges and from them construct the so called *line graph*, where the nodes are the potential $q-$step reachable edges in the original graph. These strongest edges are assumed to be *stable*. By simulating random walks on the line graph, we identify potential new edges in the original graph. This approach is fully unsupervised and it is both more scalable and robust than recent explicit spectral methods, such as the Semi-Definite Programming (SDP) densifier and the sufficient condition for decreasing the spectral gap. Experiments show that our method is only outperformed by some choices of the parameters of a related method, the anchor graph, which relies on pre-computing clusters representatives, and that the proposed method is effective on a variety of real-world datasets.

Keywords: Graph densification, Dirichlet problems, Random walkers, Commute Times

Email addresses: `manuel.curado@ua.es` (Manuel Curado), `sco@dccia.ua.es` (Francisco Escolano), `mlozano@dccia.ua.es` (Miguel A. Lozano), `edwin.hancock@york.ac.uk` (Edwin R. Hancock)

2010 MSC: 00-01, 99-00

1. Introduction

In this paper, we introduce a novel methodology for improving the estimation of commute distances in mid-size graphs, namely *Dirichlet graph densification.* Given a graph, we transform or rewire it by adding new edges so that its algebraic connectivity is better conditioned. This problem arises in graphs consisting of clusters of nodes, where links are missing within the clusters. As a result the commute distances between clusters are underestimated compared to those within clusters. In a dumbbell structure consisting of two clusters of interconnected edges and a set of bridging edges (a bottleneck) between clusters, the commute distance underestimation will cause the inter-cluster distance to shrink relative to the intra-cluster distance. The problem can be solved by introducing new intra-cluster edges, thus reducing the intra-cluster commute distances and preserving the bridge or bottleneck.

Graph densification was introduced in [1], where it is posed as a constrained optimization problem driven by cut preservation. However, the link between densification and commute times was firstly explored in [2], where we highlighted the fact that densification leads to a shrinkage of the inter-cluster distances, thus making commute times meaningful in large graphs. Later on, in [3], we highlighted the fact that state-of-the-art densifiers rely on semi-definite programming and motivate a novel algorithm, which is more scalable and robust. The core of this algorithm is harmonic analysis. Herein, we retain the basic formulation in [2] and enrich its analysis with that of another recent spectral method. Regarding [3], we: (i) clarify the use of the Dirichlet principle, (ii) relate it to the implicit minimization of the spectral gap and (iii) add a significant amount of experiments to test the proposed method in several datasets.

To develop the mathematical machinery for this study, we commence by exploring the link between the Cheeger constant [4], the spectral gap [5] and commute distances [6]. We then show the role of harmonic functions when diffusing *edge certainty or edginess* information through the so called *line graph.* This is the graph of q-step reachable edges. Since harmonic functions are consistent with the concept of smoothness, a good densifier is one that not only retains the strongest intra-cluster edges originally present but can also input new edges that fill critical intra-cluster gaps while minimizing the number of inter-cluster links. They thus improve the algebraic connectivity of the resulting graph.

In our previously published papers, we have briefly introduced and sketched the overarching concept of using graph densification as a means of pre-conditioning spectral clustering, thus motivating the need for densification in accurate clustering [2]. In this paper, we also reviewed the fundamentals of graph densifiers based on cut similarity and then analyzed the associated optimization problems, using just toy examples to provide proof of concept and to illustrate our hypothesis in the estimation of commute times. Moreover, in [3], we have proposed a specific graph densifier based on minimizing the combinatorial Dirichlet integral for the specific case of the line graph. This approach estimates meaningful commute distances for mid-sized graphs. However, it is fully bottom up and unsupervised, whereas the state-of-the-art as exemplified by anchor graphs requires a top-down and supervised approach. In this paper we therefore extend this work, presenting a more general and practical framework for graph densification and commute time estimation. This constitutes a full analysis of densification which, in contrast to our earlier work is principled, tractable and bottom-up.

Sections 2 and 3 cover some of the same ground as [2] [3], but in greater detail and with improved notation. We commence by providing a deeper mathematical background and more detailed motivation for our approach in Section 2. In Section 3, we review the mathematical detail of classical (spectral) densifiers. Our novel contributions, on the other hand, are described in the second part of the paper, i.e. in Section 4 onwards. We pose the problem in terms of applying the Dirichlet principle in Section 4. In Section 5, we explain how to apply this principle, commencing by considering the problem of how to group edges through return random walks. Return random walks (RRW) are designed to enforce intra-class edges while penalizing inter-class weights, and we reformulate our original algorithm to improve the efficiency and to capture more efficiently which links are intra-class edges, removing the noise (inter-class edges). Since our strategy is completely unsupervised, the return random walks operate under the hypothesis that inter-class edges are rare events. We consider the RRW as a filter which can be used to remove noise. To this end, in Section 6 we exploit the random walker [7] not on the original graph but on the corresponding line graph, where the nodes become the potential edges (reachable in q steps) in the original graph. We show that the random walker minimizes the Dirichlet integral, in this case that associated with the line graph. In Section 7, we have analyzed the new RRW algorithm and its resulting densification levels on a variety of data including anchor graphs. We also test the sensitivity of our approach to the two thresholds, which define universal bounds of densification and which are applicable to multiple datasets. This illustrates that the extended densification framework presented in this paper outperforms that reported in our previous work [2] [3]. Finally, we draw our conclusions and discus future work in Section 8.

2. Background

We first introduce the concept *graph densification* and its formulation as a constrained optimization problem in which cuts are to some extent preserved in the densified graph. Since this is consistent with the preservation of bottlenecks, we establish a link with the minimization of graph conductance (or Cheeger constant) Φ. Minimizing or constraining the graph conductance leads us to constrain the spectral gap λ_2, since $\lambda_2 \leq 2\Phi$. As a result, commute times between nodes in the densified graph become meaningful. This allows us to motivate the development of a scalable densifier.

2.1. Graph Densification

Graph densification [1] is the principled study of how to significantly increase the number of edges of a graph $G = (V, E)$ so that the new graph $H = (V, E')$, approximates G with respect to a given test function, for instance whether there exists a given cut within the two graphs. The study supported in this paper is motivated by the fact that certain NP-hard problems have a PTAS (Polynomial Time Approximation Scheme) when their associated graphs are dense. This is the case of the MAX-CUT problem [8]. Frieze and Kannan [9] raise the question of whether this computational ”easyness” can be explained by the Szemerédi Regularity Lemma. This lemma states that very large dense graphs have many of the properties of random graphs [10].

For a standard machine learning setting, we have that the graph G is typically sparse. This may occur for instance when either a kNN representation is used or when a Gaussian graph is constructed with a small bandwidth parameter σ. In this case the densification of G so that the value of any cut is at most C times the value of the same cut in G is called a *one-sided C-multiplicative cut approximation.* This (normalized) cut approximation must satisfy:

$$\frac{cut_H(S)}{vol(H)} \leq C \cdot \frac{cut_G(S)}{vol(G)} \ , \tag{1}$$

for any subset $S \subset V$ of the set of vertices V, where $cut_G(S) = \sum_{i \in S, j \in V \sim S} w_{ij}$ considers the set of edge weights $\{w_{ij}\}_{i,j \in V}$, where $w_{ij} \in [0,1]$. For H, we have $cut_H(S) = \sum_{i \in S, j \in V \sim S} w'_{ij}$ for edge weights $\{w'_{ij}\}_{i,j \in V}$ also satisfying $w'_{ij} \in [0,1]$. The cuts are normalized by the total edge weight $vol(.)$ of each graph, i.e. $vol(G) = \sum_{i,j} w_{ij}$ and $vol(H) = \sum_{i,j} w'_{ij}$. With these ingredients, graph densification can be posed in terms of solving the following (primal) problem:

$$\mathbf{P1} \ \ Max \ \sum_{i,j} w'_{ij} \ \ s.t. \ \ \forall S \subseteq V : \ \ \frac{cut_H(S)}{vol(H)} \leq C \cdot \frac{cut_G(S)}{vol(G)} \ , \tag{2}$$

where w'_{ij} are the weights of the new graph $H = (V, E')$, $cut_H(S)$ is the total weight of the cut induced by S, but in the new graph H instead of in the original one G. Finally $C > 0$ is a constant. The *cut-bounding constraints* are designed so that bottlenecks or bridges are preserved as much as possible (depending on C).

The preservation or enforcement of bottlenecks is key to improving the consistency of the optimal graph-based partition or clustering (see [11] and references therein). This problem is also compatible with the minimization of the graph conductance or Cheeger constant Φ [4] which is defined as

$$\Phi \triangleq \min_{S \subseteq V} \frac{cut(S)}{\min(vol(S), vol(\bar{S}))} \ , \tag{3}$$

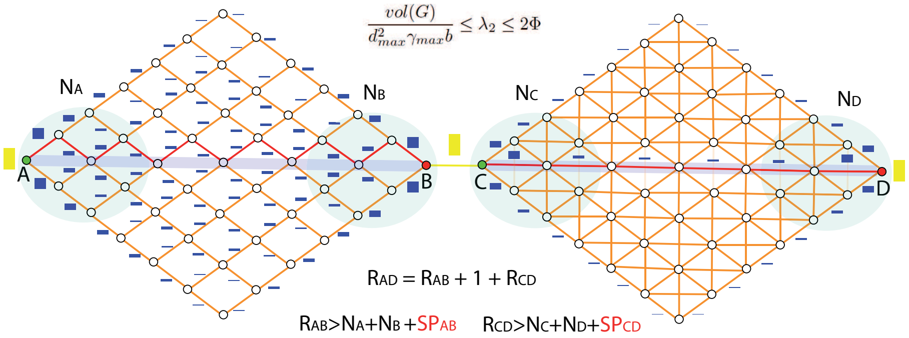
where $cut(S) = \sum_{i \in S, j \in \bar{S}} w_{ij}$ is the weight of the cut associated with S, the subset of vertices, and $vol(S) = \sum_{i \in S} d_i$ (where d_i is the degree of the node i) is the volume (density) of S which determines the density of the graph.

Since solving **P1** naturally involves increasing $vol(H)$, it leads to a succinct approximation of the bound $\Phi_H \leq C\Phi_G$, where Φ_G and Φ_H are the respective Cheeger constants of G and H. Such an approximation has an important impact in the improvement of fundamental node-to-node similarities such as Commute Times.

2.2. Densification and Commute Times

Graph densification was originally proposed as a formal tool for ruling out the existence of certain embeddings [1]. In other words, if graphs are embeddable then they cannot be densified and vice versa. However, the above observation involving Cheeger constants makes this topic more appealing for pattern recognition. Specifically, graph densification can be seen as a way of pre-conditioning or rewiring graphs so that they boost the tractability of subsequent processing tasks. One of these tasks is the measurement of the similarity between nodes.

The accurate measurement of the similarity between nodes is a key problem in graph-based learning and pattern recognition. Commute times (CTs), for instance, are Euclidean distances that rely on random walks. Namely, given a graph $G = (V, E)$ and two nodes $i, j \in V$, the commute time CT_{ij} between them is the expected time taken for a random walk to travel from i to j and back again [12][13][6]. The link with the resistance distance $R_{ij} = \frac{1}{2|E|}CT_{ij}$ characterizes the diffusive nature of commute times. If we consider unit flows between nodes, the limitations of commute times become clear. As an illustration, in Fig. 1, we show the unit flow between nodes A and D. The input unit flow (yellow bar) scatters through the left part of the graph (a 4N grid), and then it is recovered (due to the flow conservation law) at node B. In this example the flow crosses the single edge cut and then, later on it re-enters a 8N grid. Here, it diffuses until node D is reached. In the specific case of a $p = 2$ norm, this leads to the approximation $R_{ij} \approx \frac{1}{d_i} + \frac{1}{d_j}$, where d_i and d_j are the degrees of nodes i and j respectively [14].



$\frac{vol(G)}{d^2_{max}\gamma_{max}b} \leq \lambda_2 \leq 2\Phi$
NA
NB
NC
ND
A
B
C
D
RAD = RAB + 1 + RCD
RAB>NA+NB+SPAB
RCD>NC+ND+SPCD

Figure 1: Commute Times and densification: A 4N-graph linked with a 8N-graph through a single edge. Unit flow (magnitude of vertical bars) between nodes A and D. Flow is concentrated in the neighbourhoods $N_A \ldots N_D$. It diffuses through the 4N graph and then, after the cutting edge, it becomes unitary again and enters the 8N graph. In this case, flow scattering (not completely shown for the 8N graph for clarity) is symmetric with respect to the (horizontal) axis $A \rightarrow D$. Shorter paths (SP) are shown in red. Densification (8N graph) produces many more (and shorter) paths, which decreases the spectral gap λ_2. The smaller the cut in conjunction with densification, the better is the divergence of R_{AD} from $\frac{1}{d_A} + \frac{1}{d_D}$.

As a result, CTs are globally meaningless, unless we re-scale or re-define them. In [15], Luxburg et al. amplify CTs by using $S_{ij} \triangleq R_{ij} - \left(\frac{1}{d_i} + \frac{1}{d_j}\right)$. In [16], they further show that the computation of CTs relies on $p-$resistances.

These developments lead us to decompose the effective resistance in terms of a local component and a global one. Nguyen and Mamitsuka [17] propose to 1) calculate the $p = 2$ flow and 2) define a modified $p-$resistance. Returning to Fig. 1, this means that the bars are no-longer reduced and this to some extent overcomes the problem of global information loss.

In this paper, we turn our attention to *change the topology of a graph* G as a means of alleviating the above problems and providing more robust estimates of CTs, density and geodesics. Our approach, referred to as *graph densification*, implicitly minimizes the spectral gap of the input graph G since this relaxes the bounds on the local component of the CTs. We show that a) the SDP formulation is too simple to preserve global information in realistic situations, and b) SDP solvers are polynomial in the number of unknowns [18], which are $O(n^2)$ in this case, and thus only small-scale experiments can be performed. As a result, we propose a fully unsupervised approach that densifies the graph both more efficiently and more accurately than SDP. This method is referred to as *Dirichlet densification*. In the following, we show that Dirichlet densification is consistent with the minimization of the spectral gap and also that it is a better alternative than anchor graphs [19] as a method for conditioning or preprocessing the adjacency matrix (or more precisely the Laplacian matrix) for subsequent pattern recognition tasks.

2.3. The ingredients of our approach

The starting point of our approach is the following bound, derived by von Luxburg et al. [14] for any connected, undirected graph $G = (V, E)$ that is not bipartite:

$$\left|\frac{1}{vol(G)}CT_{ij} - \left(\frac{1}{d_i} + \frac{1}{d_j}\right)\right| \leq 2\left(\frac{1}{\lambda_2} + 2\right)\frac{w_{max}}{d^2_{min}} \tag{4}$$

where $CT_{ij} = R_{ij}vol(G)$ is the commute time between the nodes i and j, $vol(G)$ is the volume of the graph, λ_2 is the *spectral gap* and d_{min} is the minimum node degree in G. The spectral gap λ_2 is the second eigenvalue of the normalized graph Laplacian $\mathcal{L} = I - D^{-1/2}WD^{-1/2}$ where $D = diag(d_1, \ldots, d_n)$ is the degree matrix and W is the (symmetric) affinity (or weight) matrix, with $w_{ij} > 0$ if $(i, j) \in E$, and w_{max} is the maximal affinity element in W.

The above equation suggests that a way of making $R_{ij} \approx \frac{1}{d_i} + \frac{1}{d_j}$ diverge is to reweight/rewire the edges in E so that $\lambda_2 \to 0$. To commence, we have the following lower bound:

$$\frac{vol(G)}{d^2_{max}\gamma_{max}b} \leq \lambda_2 \ , \tag{5}$$

where γ_{max} is the maximum path length [5]. The definition of b relies on the set of paths $\{\gamma_{ij}\}$ between any pair of vertices $i \neq j$: $b \triangleq \max_{e\in E} \mathbb{E}\,|\{\gamma_{ij}\}:\; e \in \gamma_{ij}\}|$. Then, b is associated with the most traversed edge e. Actually, b is the expected number of paths traversing such an edge. Thus, Eq. 5 shows that λ_2 is minimized when $b \to \infty$, i.e. with there exists a small bottleneck defined by a handful of maximal traversed edges (see, for instance, the yellow edge in Fig 1). A single-edge cut leads to make $R_{AD} = R_{AB} + 1 + R_{CD}$ thus diverging from the resistance distance $\frac{1}{d_A} + \frac{1}{d_D}$, especially when R^1 (effective resistance with norm $p = 1$) is used. However, as soon as a small number of edges link the 4N and 8N graphs in Fig. 1 the spectral gap grows and $R_{AD} \approx \frac{1}{d_A} + \frac{1}{d_D}$.

The existence of a small bottleneck is also compatible with the minimization of the graph conductance or Cheeger constant Φ [4]. Then, we have the following upper bound for λ_2:

$$\lambda_2 \leq 2\Phi, \tag{6}$$

where Φ is the Cheeger constant (Eq 3). This bound suggests that λ_2 is minimized when: a) the cut is minimized (see above), and b) $\min(vol(S), vol(\bar{S}))$ is as large as possible. It is well known that for two cliques of size n linked by r edges, we have $\Phi = \frac{r}{n(n-1)}$, i.e. $\lim_{n\to\infty}\Phi = 0$. However, if $r = n$ the we need larger cliques for constraining the spectral gap.

This rationale opens the door to modify the set of edges E, by adding and/or reweighting edges so that $\min(vol(S), vol(\bar{S}))$ is maximized for all $S \subset V$. However, we must take into account the fact that the Cheeger constant relies on the worst case. For instance, in Fig. 2, the left cluster, say S, is less dense than the right one, $\bar{S}$. Its density (the worst case) constrains the graph conductance instead of that of $\bar{S}$. On the one hand we need to infer more edges for S, but on the other hand, we must minimize the number of new inter-class edges linking S and $\bar{S}$ (dashed links in Fig 2-right).

This could be done by solving **P1** (graph densification), but in this paper we derive a more precise and scalable densification. This new procedure is designed to *implicitly minimize the Cheeger bound* (Eq. 6) as follows. We define

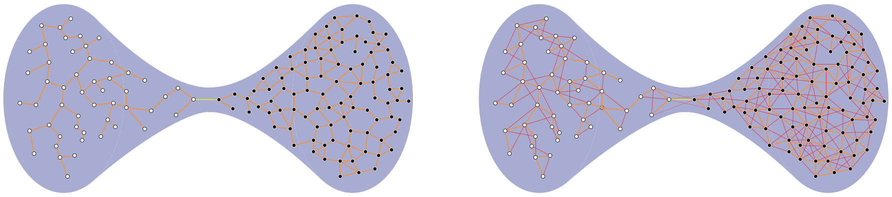


Figure 2: Cheeger constant and Densification. Left: before densification, the left (less dense) cluster, say S, conditions the Cheeger constant. In this case, $cut_G(S) = 1$ (see the extremal nodes in the yellow edge). Right: after densification, $cut_H(S) = cut_G(S) + 2d$ with $d = 2$. Again, the Cheeger constant is dominated by the left cluster which is not dense enough for making $\Phi_H < \Phi_G$.

a measure of "edginess" which is harmonically diffused. Harmonic diffusion enforces a conservative (minimum energy) method of inducing new edges [20]. However, this may be not strong enough to minimize the Cheeger bound. Let Φ_G be the (input) Cheeger bound associated with $G = (V, E)$, and $cut_G(S)$ the minimal cut in G and $S \subset V$ the set associated with this cut. If G is unweighted then $cut_H(S) = cut_G(S) + kO(d)$, where d is the average degree of the k *extremal nodes* involved in the cut. As a result, it is straightforward to see that $\Phi_H < \Phi_G$ only if $min(vol_H(S), vol_H(\bar{S})) = O(d^2)$. This means that we require a quadratic density (virtually to transform S into a clique, where $d = n - 1$) to improve the input bound. This explains why structural noise (strength of inter-class links) constrains the effectiveness of any densifier.

3. Spectral Graph Densification

3.1. Spectral Relaxation

In the previous section, we have formulated the problem of graph densification and then we have explored the implications of densifying a graph in terms of improving the measurement of certain node similarities such as the commute distance. With these similarities to hand, certain subsequent processes such as graph-based clustering and ranking, and even graph compression, can be improved. Thus, graph densification (or more generally *graph rewiring*) is herein conceived as a *structural filter*. However, to that end we have to propose both scalable and accurate methods.

We commence by highlighting the combinatorial nature of the problem as well as its implications. Following [1], the formulation in Eq. 2 is equivalent to

$$\begin{array}{lll} \mathbf{P1} & Max & \sum_{i,j} w'_{ij} \\ s.t. & \forall\, i,j: & w'_{ij} \leq 1 \\ & \forall\, S \subseteq V: & \sum_{i\in S, j\in V\sim S} w'_{ij} \leq C \cdot cut_G(S) \sum_{i,j} w'_{ij} \\ & & w'_{ij} \geq 0\ , \end{array} \tag{7}$$

since $cut_H(S) = \sum_{i\in S, j\in V\sim S} w'_{ij}$ and $vol(H) = \sum_{i,j} w'_{ij}$. Herein, we simply drop the normalization constant $vol(G)$ in each constraint (formally, C absorbs this normalization). Obviously, $\mathbf{P1}$ has $O(2^n)$ constraints since the weights w'_{ij} in H are maximized s.t. all the possible normalized cuts in H (one per each $S \subseteq V$) are constrained by their homologs in G. Thus, with spectral relaxation to hand one can replace all these constrains by a unique (matricial) one.

If z is the binary $0-1$ characteristic vector of a given $S \subseteq V$ then

$$\begin{array}{rcl} cut_G(S) & = & z^T L_G z \triangleq \sum_{(i,j)\in E} w_{ij}(z_i - z_j)^2 \\ cut_H(S) & = & z^T L_H z \triangleq \sum_{(i',j')\in E'} w'_{ij}(z_i - z_j)^2\ , \end{array} \tag{8}$$

where $L_G = D_G - W$ and $L_H = D_H - W'$ are the respective Laplacian matrices of $G = (V, E)$ and $H = (V, E')$, with diagonal degree matrices D_G, D_H and weight matrices W and W'. In addition, if H satisfies

$$z^T L_H z \leq C \cdot z^T L_G z\ , \tag{9}$$

for any $z \in \mathbb{R}^n$, with $n = |V|$, we say that G and H are $C-$*spectrally similar* and this fact is denoted by $L_H \preceq C \cdot L_G$. Spectrally similar graphs share many algebraic properties [21]. For instance, their effective resistances (rescaled commute times) are similar. This similarity is bounded by C and it leads to nice interlacing properties. We have that the eigenvalues of $\lambda_1, \ldots, \lambda_n$ of L_G

and the eigenvalues $\lambda'_1, \ldots, \lambda'_n$ of L_H satisfy: $\lambda'_i \leq C \cdot \lambda_i$. This implies that, for $C \geq 1$, H does not necessarily modify the spectral gap of G and the eigenvalues of L_G are not necessarily shifted (i.e. increased).

However, spectral similarity provides a way of replacing a exponential number of constraints by a unique constraint $L_H \preceq C \cdot L_G$, thus enforcing spectral similarity. Then, if $b_{ij} = e_i - e_j$, where e_i is the vector with all zeros but a 1 in the $i-$th position (and similarly for e_j), we have $L_H = \sum_{i,j} w'_{ij} b_{ij} b_{ij}^T$ and **P1** can be relaxed using Semi-definite programming (SDP) as follows

$$\begin{array}{lll} \mathbf{P1_{SDP}} & Max & \displaystyle\sum_{i,j} w'_{ij} \\ s.t. & \forall\, i,j: & w'_{ij} \leq 1 \\ & & \displaystyle\sum_{i,j} w'_{ij} b_{ij} b_{ij}^T \preceq \left(C \cdot \sum_{i,j} w'_{ij} \right) L_G \\ & & w'_{ij} \geq 0\,. \end{array} \tag{10}$$

In practice, this problem is better approached by its dual one $\mathbf{P2_{SDP}}$ which implicitly seeks a proper embedding for the nodes of V. Summarizing, the optimal embedding is encoded in the columns of a $n \times n$ positive semi-definite matrix $Z \succeq 0$. Given these coordinates and the optimal values of the dual variables σ_{ij} we can obtain w'_{ij} (see details in [2]).

3.2. Testing Densification with Toy Experiments

With the primal SDP problem $\mathbf{P1_{SDP}}$ to hand we have that

$$\lambda'_i \leq \left(C \cdot \sum_{i,j} w'_{ij} \right) \lambda_i \tag{11}$$

where λ'_i are the eigenvalues of the Laplacian L_H associated with the densified graph H. For $C > 1$ we have that densification tends to produce a complete graph $\mathcal{K}_n$. When we add to the cost of the dual problem $\mathbf{P2_{SDP}}$ the term $-K \log\det(Z)$ (a log-barrier), it enforces choices for $Z \succeq 0$ (i.e. ellipsoids) with maximal volume which also avoids the complete graph. In this way, given a fixed $K = 1000$, the structure of the pattern space emerges[1] as we modify the $C < 1$ bound so that the spectral gap is minimized in such a way that reasonable estimates of the commute distance emerge.

Synthetic Experiments. We have designed several instances of the *double moon* configuration of points following [23]. The dataset consists of two regions A and B representing two classes. Each region is a half annulus with radius $r = 10$, width $w = 6$. Region A is upper, most centered at $(0, 0)$ and region B is lower, most centered at (r, d), where d is the distance, and mirroring A with respect to the x axis. If $d < 0$ then some overlap is assumed. The more negative is d the higher is the overlap. We consider three levels of overlap: $d = -1$, $d = -2$ and $d = -3$. Once we have constructed a Gaussian proximity graph for each level, and chosen a C-bound constant, we: (1) apply the SDP densifier, (2) estimate commute times from the result, and (3) measure the Adjusted Rand Index (ARI), as a percentage, with respect to the ground truth. Then, for $d = -1$ we obtain ARI 98% for $C = 0.1$ and 100% for $C = 0.025$. For $d = -2$, $C = 0.1$ yields 84.5%, and 100% is obtained for $C = 0.025$. However, for $d = -3$ (large overlap), setting $C = 0.025$ leads to an ARI of only 20.5%, and we must relax C to 0.00625 to achieve the best ARI for this instance, i.e. 96%. In all these experiments, we set $n = 100$.

Real Datasets (NIST). We subsample the NIST handwritten digit dataset[2] and build Gaussian graphs to construct challenging instances for the SDP densifier: 2-classes (digits 5 and 6), 4-classes (digits 3, 5 and 7), 3-classes (digits 5 to 8), 5-classes and 10-classes. In all cases, $n = 100$ and all classes have the same number of samples. We investigate the C bound in the range $C \in [0.05, 0.75]$. In this case, we obtain large ARIs for small values of C in some cases (small number of classes), but the maximum ARI is 8% (for 2-classes). This indicates

[1] All examples/experiments in this section were obtained with the SDPT3 solver [22] version 4.0. In our experiments, the number of variables is $|E| \approx 4500$ and the SDP solver is polynomial in $|E|$.

[2] http://yann.lecun.com/exdb/mnist/

that in real world datasets, where more structured inter-class noise arises, the quality of the results is highly conditioned by the simplicity of the optimization problem (guided only by a *blind* spectral similarity, which does not necessarily reduce inter-class noise).

The above experiments show that the SDP formulation is neither effective nor scalable. However, the fact that SDP address an explicity minimization of the spectral gap (see Eq. 11) suggests to review alternative, yet spectral methods, for performing such a minimization.

3.3. Sufficient Conditions for Decreasing the Spectral Gap

Given the original graph $G = (V, E)$, let $\mathcal{L} = I - D^{-1/2}WD^{-1/2}$ be its normalized Laplacian matrix. Then, the spectral gap λ_2 can be posed in terms of the eigenvalue f (Fiedler vector) associated with the spectral gap, as follows [24]:

$$\lambda_2 = \min_{f \perp D^{1/2}1} \frac{\sum_{(i,j)\in E} w_{ij}(f(i) - f(j))^2}{\sum_{j>i}(f(i) - f(j))^2} \tag{12}$$

Let $H = (V, E')$ with $E' = E \bigcup \{(i, j)\}$ be the graph obtained by adding the edge $(i, j) \notin E$ to G. Eldan et al., have recently derived a sufficient condition for achieving $\lambda_2(H) < \lambda_2(G)$ after including (i, j) (see Lemma 1 in [25]). After some algebraic manipulations, the resulting condition is

$$\frac{\lambda_2(G)}{vol(G)}\left(\frac{1}{2\sqrt{d_i}} + \frac{1}{2\sqrt{d_j}}\right)^2 + (1 - \lambda_2(G))\left\{\frac{f^2(i)}{d_i} + \frac{f^2(j)}{d_j}\right\} < \frac{f(i)f(j)}{\sqrt{d_i}\sqrt{d_j}}\ , \tag{13}$$

Therefore, it is possible to predict whether including a new edge leads to decrease the spectral gap in H simply by using the Fiedler vector of G (assumed to have unit norm in the following). More precisely, for a setting with two clusters, we have that $f(i)f(j) < 0$ if (i, j) is an inter-class link, due to the structure of the Fiedler vector [26]. Since inter-class links lead to increase the spectral gap, the above condition is not applicable. When $f(i)f(j) > 0$ and we have two clusters, we know that (i, j) is an intra-class link and it should be included since the spectral gap is reduced for sure. Consequently, the above equation is useful when we have more than two clusters. For instance, for 3 clusters, f has three types of entries $\{+k, r \rightarrow 0, -k\}$ which are respectively assigned to the nodes of the corresponding clusters. In this case, we may have $f(i)f(j) > 0$ for inter-class links, such as $f(i) = |k|$, $f(j) = r$. However, now $k^2 + r^2 \gg |k| \cdot r$ and the condition in Eq. 13 is only satisfied when $d_i \gg d_j$. A similar requirement is needed when $f(i) = f(j) = k$ or $f(i) = f(j) = r$. As a result, this condition enforces preferential attachent (link nodes with a small degree to those with large degrees) but it does not tell us (in general) whether the edges satisfying the condition are either inter-class edges or intra-class edges.

4. Towards Dirichlet Densifiers

4.1. Implicit Minimization of the Spectral Gap

Since spectral methods (SPD and sufficient conditions) have severe limitations to provide both scalable and reliable densifiers, we turn our attention to the implicit minimization of the spectral gap. In Section 2.3, we exploited the link between the spectral gap λ_2 and graph conductance Φ ($\lambda_2 \leq 2\Phi$) to pose densification in terms of adding edges to the original graph $G = (V, E)$ so that Φ is significantly bounded. In practice, where G is attributed, we assume that intra-class weights are generally smaller than inter-class ones. Under this condition, we can rely on large-valued w_{ij}s to *diffuse edginesss in a conservative way* (minimum risk of increasing the spectral gap). To that end, let $z \in \{0, 1\}^{\mathcal{E}}$, be an indicator vector where $\mathcal{E} \subseteq |V| \times |V|$ is the set of edges in E (with $w_{ij} > 0$) sharing a vertex in V. Two *neighbouring edges* $e_a = (i, k)$ and $e_b = (k, j)$ share a vertex k. If we define $w(e_{ab})$ as a similarity measure between two neighbouring edges, for instance $w(e_{ab}) = w_a \cdot w_b$ or $w(e_{ab}) = \min(w_a, w_b)$, the most conservative way of difussing edginess is given by the Dirichlet principle.

Such a principle consists of defining z as a minimizer of

$$Q(z) = \sum_{e_a \sim e_b \in \mathcal{E}} w(e_{ab})(z_a - z_b)^2 \ , \tag{14}$$

which leads to

$$z_a = \frac{1}{d_a} \sum_{e_a \sim e_b \in \mathcal{E}} w(e_{ab}) z_b \ , \tag{15}$$

where $d_a = \sum_{e_b \sim e_a} w(e_{ab})$. Consequently, the Dirichlet principle leads to diffuse edginess in an harmonic way (the edginess of a given edge is the weighted average of those of its neigbouring edges). Since large-valued edges are assumed to be the most confident ones, we set *some* z_bs to 1 in the above equation to infer the *unknown* z_as, thus relaxing the domain $\{0,1\}^{\mathcal{E}}$ to $[0,1]^{\mathcal{E}}$ for z.

4.2. A Toy Example

In order to illustrate the Dirichlet principle, given a small graph $G = (V, E)$ like the one in Fig. 3-top, let the weight of each of its edges be proportional to the thickness of the lines linking the corresponding vertices. In this regard, dashed lines mean some potential new edges resulting from densification. The green one is given by edges $(a3, a0)$ and $(a0, b0)$, i.e. it is an inter-class edge. However, the blue one is given by $(b1, b0)$ and $(b0, b3)$ and it is an intra-class edge. Then, the diffusive process rooted in the Dirichlet principle is applied to the *line graph* whose nodes are the edges in E and its edges $\mathcal{E}$ are given by two-step node transitivities (or, equivalently, one-step edge transitivities) in G. Any edge in the line graph is candidate for densifying G, but these edges must be discovered by the Dirichlet diffusion process. If this process starts at $a0 - b0$ (the green node) it will reach four inter-class edges (now nodes in the line graph) and the spectral gap will increase. On the other hand, the diffusion process is optimally *seeded* at $a1 - a2$ and $b2 - b3$ (red nodes) which strongly link intra-class nodes in G. An interesting property of the line graph is that there are more intra-class nodes than inter-class ones. For instance, the intra-class edge $b1 - b3$ can be inferred either from the path $b2 - b3, b1 - b2$ or from the path $b0 - b3, b0 - b1$. If we start the process by setting $z_{b2-b3} = 1$ and zero elsewhere, we will find a large value for z_{b1-b2} in only one iteration, thus inferring $b1 - b3$.

In the following sections we will: (1) develop an structural filter using *return random walks* so that we can pre-filter inter-class edges in the original graph, (2) construct the line graph by considering the practical (spatial) limitations, (3) run the Dirichlet process and (4) test it in real mid-size/large graphs.

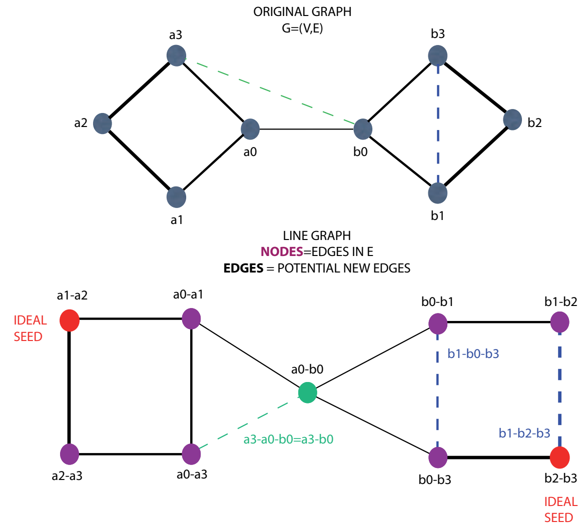


Figure 3: Dirichlet principle. Input graph with some inferred edges (top). Line graph with seeds (bottom).

5. Return Random Walks

5.1. Motivation

Our proposed densifier infers new intra-class edges while minimizing the number of new inter-class edges. To this end, we proceed to 1) design an structural filter, using Return Random Walks (RRW) and 2) build the line graph and run a Dirichlet process on it. In this section, we show that the RRWs implement a weighted diffusion process. This process minimizes the probability that a random walk starting and ending at a given node traverses the inter-class links. The resulting weighting matrix W_e is denser and more clustered than that associated with input graph.

5.2. Design of Return Random Walks

Given a set of points $\chi = \{\vec{x}_1, ..., \vec{x}_n\} \subset \mathbb{R}^D$, we map the points $\vec{x}_i$ to the vertices V of an undirected weighted graph $G(V, E, W)$. We have that V is the set of nodes where each v_i represents a data point x_i, and $E \subseteq V \times V$ is the set of edges linking adjacent nodes. An edge $e = (i, j)$ with $i, j \in V$, exists if

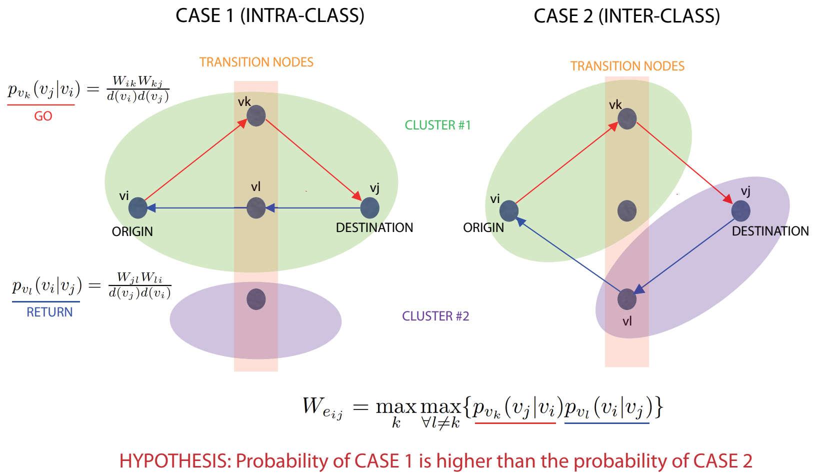

CASE 1 (INTRA-CLASS)
CASE 2 (INTER-CLASS)
TRANSITION NODES
TRANSITION NODES
$p_{v_k}(v_j|v_i) = \frac{W_{ik}W_{kj}}{d(v_i)d(v_j)}$
GO
vk
CLUSTER #1
vi
vl
vj
ORIGIN
DESTINATION
vk
vi
vj
ORIGIN
DESTINATION
vl
$p_{v_l}(v_i|v_j) = \frac{W_{jl}W_{li}}{d(v_j)d(v_i)}$
RETURN
CLUSTER #2
$W_{e_{ij}} = \max_k \max_{\forall l \neq k} \{\underline{p_{v_k}(v_j|v_i)}\underline{p_{v_l}(v_i|v_j)}\}$
HYPOTHESIS: Probability of CASE 1 is higher than the probability of CASE 2

Figure 4: Return random walks for reducing inter-class noise.

$w_{ij} > 0$, where $w_{ij} = e^{-||\vec{x}_i - \vec{x}_j||^2/\sigma^2}$, and $j \in N_k(i)$ (j is a kNN of i). The bandwidth parameter σ is optimally selected with respect to k.

Design of W_e. Given $W = \{w_{ij}\} \in \mathbb{R}^{n \times n}$, we produce a reweighted similarity matrix W_e by following the following rationale, a) we explore the two-step random walks reaching a node v_j from node v_i through any transition node v_k, b) *on return* from v_j to v_i, we maximize the probability of returning through a different transition node $v_l \neq v_k$. For the first step (going from v_i to v_j through v_k) we have $p_{v_k}(v_j|v_i) = \frac{w_{ik}w_{kj}}{d(v_i)d(v_j)}$ as well as a *standard return* $p_{v_l}(v_i|v_j) = \frac{w_{jl}w_{li}}{d(v_j)d(v_i)}$. The standard return works well if v_i and v_j belong to the same cluster (see Fig. 4-left). However, v_l (the transition node for returning) can be constrained so that $v_l \neq v_k$. In this way, travelling out of a class is penalized since the walker must choose a different path, which in turn is hard to find on average. Therefore, we obtain $w_{e_{ij}}$ from w_{ij} as follows:

$$w_{e_{ij}} = \max_k \max_{\forall l \neq k} \{p_{v_k}(v_j|v_i)p_{v_l}(v_i|v_j)\} \ , \tag{16}$$

i.e. for each possible transition node v_k we compute the probability of leaving and returning (product of independent probabilities) through a different node v_l. We retain the maximum product of probabilities for each v_l referred to a given k and finally we retain the supremum of these maxima. As a result, when inter-class paths are frequent for a given edge $e = (i, j)$ (Fig. 4-right) its weight $w_{e_{ij}}$ is significantly reduced. The weights $w_{e_{ij}}$ measure the connectivity between two nodes in a specific cluster or region (not the direct connection but the indirect one through neighbouring nodes). Large values of $w_{e_{ij}}$ mean that both nodes i and j are not only strongly locally connected but they also belong to a highly cohesive connected component.

Our working hypothesis is that the number of edges involved in inter-class transitions (Fig. 4-right) is small on average, since the number of inter-class edges tends to be small compared with the total number of edges. In realistic situations patterns can be confused either due to their intrinsic similarity or due to the use of an improper similarity measure. As a result, this assumption leads to a significant decrease of many of the elements of W.

Filtering of W_e. To reduce inter-class noise, we consider the relationship between the shortest path and the sum of different weights of the RRW, i.e.

$$w'_{e_{ij}} = w_{e_{ij}} \times \exp\left\{ -\frac{\gamma^{ij}_{min}}{\gamma_{ij}} \right\} \ , \tag{17}$$

where $\gamma_{ij} = w_{ik} + w_{kj} + w_{jl} + w_{li}$ and γ^{ij}_{min} is the length of the shortest path between i and j. Consequently, we enforce that the length of the actual path (constrained to pass through l and k with $l \neq k$) is very large in comparison with that of the shortest path between i and j. In addition, we enforce that the length of the shortest path γ^{ij}_{min} is small too.

However, the above equation does not account for the difference between outward and return paths. For this case, we assign the weight as

$$w''_{e_{ij}} = \frac{w'_{e_{ij}}}{bs_{ij}}, \ \text{where} \ bs_{ij} = \frac{w_{ik} + w_{kj}}{w_{jl} + w_{li}} \ , \tag{18}$$

where bs_{ij} measures the balance or symmetry with respect to outward and return paths (asymmetric if $bs_{ij} \neq 1$). If the value bs_{ij} is either small or large, then (i, j) will be considered an inter-class edge.

The above filtering of W is quite effective for reducing inter-class noise. In Figure 5 we show the Adjusted Rand Indices (ARIs) obtained for W_e, W'_e and

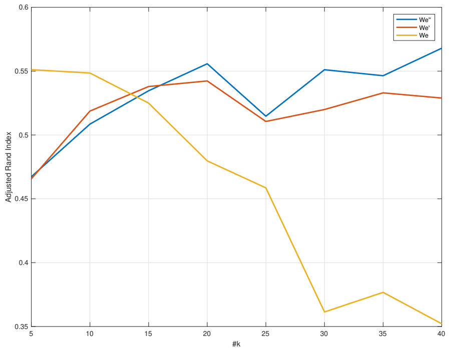


Figure 5: NIST dataset: Comparison and evolution of Return Random Walks for increasing values of k in kNN

W''_e (NIST dataset with $n = 1000$). As k increases, both W'_e and W''_e are stable, whereas the effectiveness of the kNN graph, filtered with Eq. 16, decays with k.

In the following section, we rename W''_e as W_e, to avoid notational conflicts.

6. The Dirichlet Graph Densifier

6.1. Motivation

Once we obtain the filtered weighting matrix W_e which is both denser and better conditioned than the original input matrix W, we 1) construct an oracle to create new edges and 2) run a Dirichlet process. The main novelty of our contribution is that we run random walkers on a *line graph*, i.e. in the edge space rather than the original graph. In addition, the two thresholds δ_1 and δ_2, used in the proposed approach, become universal bounds which are applicable (see details in the experimental section 7) to multiple datasets.

6.2. The Line Graph

The graph densification problem can be posed as follows: given a graph $G = (V, E, W)$ infer another graph $H = (V, E', W')$ so that $|E'| \geq |E|$ in such a way that the bulk of new edges are constrained to be intra-class edges (i.e. the number of inter-class edges is minimized). Therefore, the unknowns of the problem are the new edges that need to be inferred. In principle we have a $O(n^2)$ unknowns, where $n = |V|$, and working with all of them is infeasible. This motivates the selection of a small fraction of the original edges (those with the largest values of $w_{e_{ij}}$) according to a given threshold δ_1. This leads to an oracle $E'' = \{e \in E : w_e \geq \delta_1\}$ containing the most likely candidate edges. The fact that the smaller the latter fraction the better the accuracy seems counterintuitive, and is both explained below and explored in more detail later in the experimental section of this paper. The impact of this choice on efficiency is that only $|E''|$ edges, with $|E''| \ll |E|$, are considered for constructing a graph of edges, i.e. a *line graph* $Line_{W_e}$, as follows.

Let A be the $p \times n$ edge-node incidence matrix defined as follows:

$$A_{e_{ij}v_k} = \begin{cases} +1 & \text{if } i = k, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

Then, the adjacency matrix of $q-$steps (with $q = 2$) transitive edges is $C = AA^T - 2I_p$, where I_p is the $p \times p$ identity matrix. This is the adjacency matrix of the unweighted line graph, where the nodes e_a are given by all the possible pairs of $r = |E''|$ edges with a common vertex according to A. The edges of C indicate second-order interactions between nodes in the original graph represented by A. However, C is still unattributed (although conditioned by W_e). A proper weighting for this graph is to use standard "go and return" random walks, which gives elements of the weighted adjacency matrix as

$$Line_{W_e}(e_a, e_b) = \sum_{k=1}^{r} p_{e_k}(e_b|e_a)p_{e_k}(e_a|e_b) \;, \tag{20}$$

i.e. return walks are not applied because they become too restrictive. There is an edge in the line graph for every pair (e_a, e_b) with $Line_{W_e}(e_a, e_b) > 0$. We denote the set of edges of the line graph by E_{Line}.

6.3. The Dirichlet Functional for the Line Graph

Given the line graph $Line_{W_e}$ with r nodes (now *edges*) many of them will be highly informative according to W_e and the application of Eq. 20. We retain a fraction of them (again, those with the largest values of W_e) according to a second threshold $\delta_2 > \delta_1$. This second threshold must be set as small as possible since it defines the difference between the "known" (stable) and the "unknown" (unstable) edges. More precisely, W_e acts as a function $W_e : |E''| \rightarrow \mathbb{R}$ so that the larger its value, the more certain or trustable is a given edge as a candidate stable or known edge in the original graph G. Unknown edges are assumed to have small values of W_e and this is why they are not selected, since the purpose of our method is to infer them.

This is a classical inference problem (now in the space of edges and completely unsupervised[3]) which has been posed in terms of minimizing the disagreements between the weights of existing (*assumed* to be "known") edges and those of the "unknown" or inferred ones. In this regard, since unknown edges are typically neighbours of known ones, the minimization of this disagreement is naturally expressed in terms of finding a harmonic function. Harmonic functions $u(.)$ satisfy the condition $\nabla^2 u = 0$ which in our discrete setting leads to the following property

$$u(e_a) = \frac{1}{d(e_a)} \sum_{(e_a,e_b)\in E_{Line}} Line_{W_e}(e_a, e_b) u(e_b) \ , \tag{21}$$

The harmonic function $u(.)$ is constrained, since it is known for some values of the domain (the *perimeter* or *border*). In our case, we set $u(e_a) = w_{e_a} / \max\{w_{e_a}\}$ for $e_a \in E_B$, referred to as *perimeter or border nodes* since they are associated with assumed known edges. The harmonic function is unknown for $e_b \in E_I = E'' \sim E_B$ (the *interior nodes*). Then, finding an harmonic function given its boundary values is called the *Dirichlet problem* and it is typically formulated in terms of minimizing the following integral

$$D[u] = \frac{1}{2} \int_{\Omega} |\nabla u|^2 d\Omega, \tag{22}$$

[3] Our experiments show that δ_1,δ_2 are highly consistent for different datasets, which is one of the key contributions of this paper.

where Ω is the field. Its discrete version relies on the graph Laplacian [7] (in this case on the Laplacian of the line graph):

$$\begin{aligned} D_{Line}[u] &= \frac{1}{2}(A'u)^T P(A'u) = \frac{1}{2}u^T \mathcal{L}_{Line} u \\ &= \frac{1}{2}\sum_{(e_a,e_b)\in E_{Line}} Line_{W_e}(e_a,e_b)(u(e_a)-u(e_b))^2 \ , \end{aligned} \tag{23}$$

where A' is the $|E"| \times |E_{Line}|$ incidence matrix, P is the $|E_{Line}| \times |E_{Line}|$ diagonal constitutive matrix containing all the weights of the edges in the line graph, and $\mathcal{L}_{Line} = D_{Line} - Line_{W_e}$ with $D_{Line} = diag(d(e_a) \dots d(e_{|E"|}))$, where $d(e_a) = \sum_{e_b \neq e_a} Line_{W_e}(e_a, e_b)$, is the diagonal degree matrix. Then, $\mathcal{L}_{Line}$ is the Laplacian of the line graph.

Given the line graph Laplacian $\mathcal{L}_{Line}$ and the Dirichlet combinatorial integral D_{Line} we have that the nodes in the line graph are partitioned in two classes: "perimeter" E_B and "interior" E_I, i.e. $E" = E_B \cup E_I$. This partition leads to a reordering of the harmonic function $u = [u_B \ u_I]$ as well as the Dirichlet integral:

$$D\begin{bmatrix} u_I \end{bmatrix} = \tfrac{1}{2}\begin{bmatrix} u_B^T & u_I^T \end{bmatrix} \begin{bmatrix} L_B & K \\ K^T & L_I \end{bmatrix} \begin{bmatrix} u_B \\ u_I \end{bmatrix} \tag{24}$$

where $D\begin{bmatrix} u_I \end{bmatrix} = \frac{1}{2}(u_B^T L_B u_B + 2u_I^T K^T U_B + u_I^T L_I u_I)$ and differentiating w.r.t. u_I leads to a solution of the linear system which relates u_I with u_B:

$$L_I u_I = -K^T u_B \ . \tag{25}$$

Let $s \in [0,1]$ be a label indicating to what extent a given node of the line graph (an edge in the original graph) is relevant. We define a potential function $Q : E_B \to [0,1]$ so that for a known node $e_a \in E_B$ we assign a label s, i.e. $Q(e_a) = s$. This leads to declaring the following vector for each label:

$$m_a^s = \begin{cases} \frac{w_{e_a}}{\max_{e_b \in E"}\{w_{e_b}\}} & \text{if } Q(e_a) = s, \\ 0 & \text{if } Q(e_a) \neq s \end{cases} \ . \tag{26}$$

Finally, the linear system is posed in terms of how the known labels predict the unknown ones, placed in the vector u, as follows:

$$L_I u^s = -K^T m^s \ . \tag{27}$$

If we consider simultaneously all labels instead of just a single one, we have the solution

$$L_I U = -K^T M \ \Rightarrow U = (-K^T M) L_I^{-1} \ , \tag{28}$$

where U is a stochastic matrix with $|E_I|$ rows (one per unknown/interior edge, *to be solved*) and M has $|E_B|$ rows and columns. Then, let U_k be the $k-$th row, i.e. the probabilities that a given unknown e_k edge is compatible with any of the known edges (these probabilities have unit sum). The edginess of e_k is now given by the maximum row probability. If $e_k = (i,j)$, with $i, j \in V$ in the original graph $G = (V, E, W)$, then an edge exist in $H = (V, E', W')$ if $H_{ij} > 0$, where

$$H_{ij} = \begin{cases} \max_{e_k \in U} U_k & \text{if } e_k \in E_I \\ M_{ij} & \text{if } e_k \in E_B, \\ 0 & \text{otherwise} \end{cases} \tag{29}$$

In this way, the new edges E' are inferred by the Dirichlet process.

7. Experiments and Discussion

In our experiments, we evaluate our Dirichlet densifier on four standard datasets, namely (1) a reduced version of the NIST handwritten digits dataset: $n = 1000$ (100 samples per class - 10 classes), (2) the COIL-20 dataset[4] with $n = 1440$ (72 samples per class - 20 classes [27]), (3) the FlickrLOGOs-32 dataset[5] with $n = 2240$ (70 samples per class - 32 classes [28]) and (3) the YALE-Faces dataset[6] with $n = 2414$ (variable number of samples per class - 38 classes [29]).

Once the associated kNN graphs are densified, we estimate commute times through the Nguyen and Mamitsuka [17] method (state-of-the-art). Then, the Adjusted Rand Index (ARI) with respect to the ground truth is used to measure the performance of the densification.

[4] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[5] http://www.multimedia-computing.de/flickrlogos/

[6] http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html

Our aim is to investigate the behavior of the proposed densifier as, (1) k, the number of nearest neighbours in the kNN graph, increases, (2) the size of the oracle (and thus of the line graph) $|E''|$ (set according to threshold δ_1) increases, and (3) the number of known labels $|E_B|$ (set according to threshold δ_2) increases. Our objective is to find *universal* bounds with are commonly applicable to all the datasets.

7.1. Evaluation of Dirichlet densification in kNN graphs

To commence, we analyze the original affinity matrix W with kNNs, where the number of nearest neighbours $k \in \{15, 25, 35\}$. As k increases, the graph structure becomes very noisy (many inter-class edges appear). However, for small values of k, only the strongest classes remain and the remainder of the structure is weakened. In Figure 6, we show these effects for the NIST dataset. The respective Adjusted Rand Indices (ARIs) after estimating commute times are 69.25%, 65.62% and 63.74%.

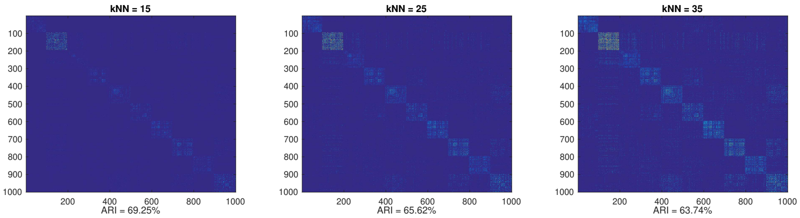


Figure 6: NIST kNN graphs for different values of k.

RRWs are critical. In Figure 7, we confirm that the performance of kNNs graphs degrades faster than that of our densification procedure. Given an input kNN graph, we apply Return Random Walks (RRWs). After RRWs, the densification level (DL) increases significantly (3.47% → 17.3%) because these walks explore all paths between nodes that are linked in $q = 2$ steps. Then, we create the line graph from the largest valued edges ($|E"|$). Finally, we obtain a densification matrix from a given percentage of known edges $|E_B|$ in order to drive the Dirichlet process. We show that the best densification level (8.22%)

with ARI (71.02%) is obtained when the fraction of selected edges for building the oracle is $|E”| = 0.35$ (δ_1 is adjusted).

In addition to their densifying role, RRWs are critical for obtaining a good ARI measure for the clusters. Given the same oracle size $|E”| = 0.35$, we obtain ARI $= 71.02\%$ using RRWs to compare to ARI $= 61.13\%$ without using them. However, the computational cost of RRWs is $O(n^4)$, and this fact must be considered in practice.

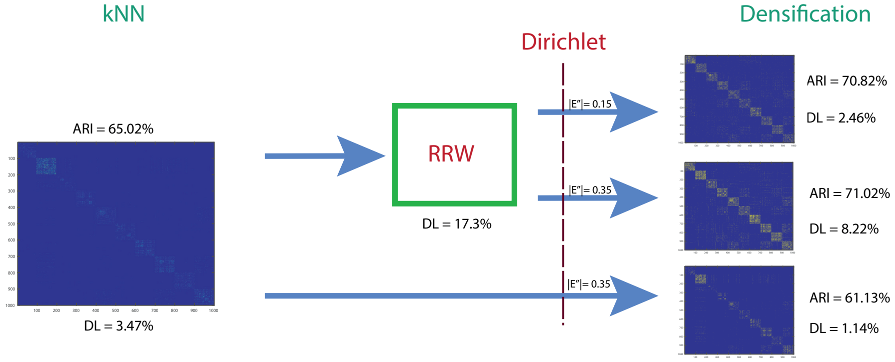


Figure 7: The Dirichlet Densifier at work (best case for NIST). We show both ARIs and DLs. Top: RRW + $|E”| = 0.15$. Middle: RRW + $|E”| = 0.35$ (best densification and ARI). Botton: not RRW but Dirichlet with $|E”| = 0.35$.

Searching for optimal thresholds. In Figure 8 we show the effect of setting the set of known edges $|E_B|$ (by adjusting δ_2) in addition to setting the fraction of edges $|E”|$ for constructing the oracle in the NIST dataset. In general, we obtain the best results for $|E”| = 0.35$ (35% edges of the original Laplacian) and a small percentage of known edges $|E_B|$ (bottom-left).

With these parameters to hand ($|E”| \approx 0.35$, $|E_B| \approx 0.05$) we analyze the four datasets with k =15, 25 and 35. The respective results are shown in Tables 1 (NIST), 2 (COIL), 3 (LOGO) and 4 (YALE). In these tables, we compare the ARI (after densification and commute times estimation) for different configurations (defined by the values of k, $|E”|$ and $|E_B|$) as well as for kNN graphs without densification. These tables should be read as follows.

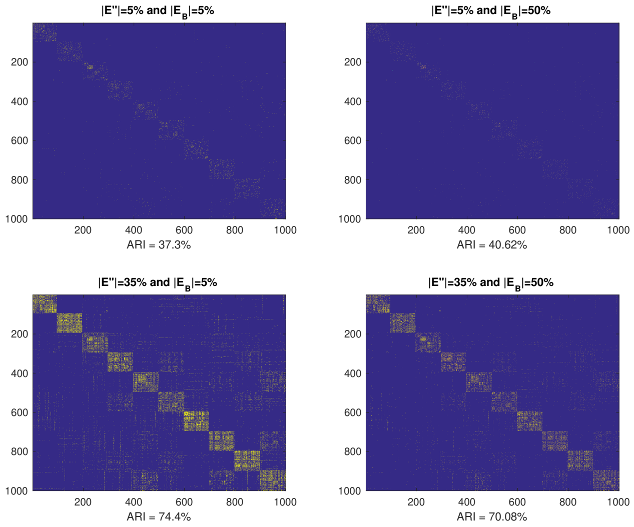


Figure 8: Densified NIST graphs with different thresholds $|E''|$ and $|E_B|$.

For each kNN scenario, the top row indicates $|E_B|$=0.05, 0.25 or 0.5 (one per column). The next block of rows corresponds to $|E"|$=0.05, 0.15, 0.25 or 0.35. The ARI is in between (in percentages). Finally, the bottom row corresponds to ARI for no densification (*No dense*).

First, the performance of raw (undensified) kNNs degrades as k increases. The magnitude of these ARI performances and their degradation rate indicates how difficult is to densify the corresponding dataset.

For the best cases, their respective ARI are 74.4% (NIST), 95.44% (COIL), 62.96% (LOGO) and 15.68% (YALE), and improve over the kNNs without densification. The corresponding results without densification are 69.25%, 89.75%, 61.92% and 14.85% respectively (see Figure 9).

Setting $|E"|$. Here, we retain a third of the edges for the oracle (35%). This is the *critical mass*, i.e. base number of representative (confident) edges, of our approach. A larger oracle may lead to a poorer result (in NIST, a kNN with k=15 and $|E"| = 0.5$ yields ARI = 71.89%) because, in these conditions we tend to include many inter-class edges (noise) in the oracle. Only in the YALE dataset, which is clearly the least structured dataset tested in this paper, the *critical mass* is reduced to $|E"| = 0.25$ to give the best performance in almost all scenarios (see Table 4).

Setting $|E_B|$. When we keep the number of known edges small ($|E_B|$=5%) the densification may become more noisy. This is due to the leakage of random walkers through inter-class edges. However (as we can see in Figure 8) we obtain denser classes (inter-class cohesiveness) in comparison to configurations with a larger number of known edges ($|E_B|$=50%). This is consistent with the reduction of the densification level (DL) as $|E_B|$ increases. For example, in the NIST dataset we have DL = 3.79% with 5% of known edges, vs $DL = 1.99\%$ with 50% of known edges.

In general, the setting $|E_B| = 0.05$ results in some loss of stability (in terms of providing optimal performance) in combination with $|E"| = 0.35$, as the kNN scenario gets harder. Otherwise, it seems more reasonable to increase the number of known labels in harder scenarios or in all the scenarios for hard/unstructured datasets (YALE).

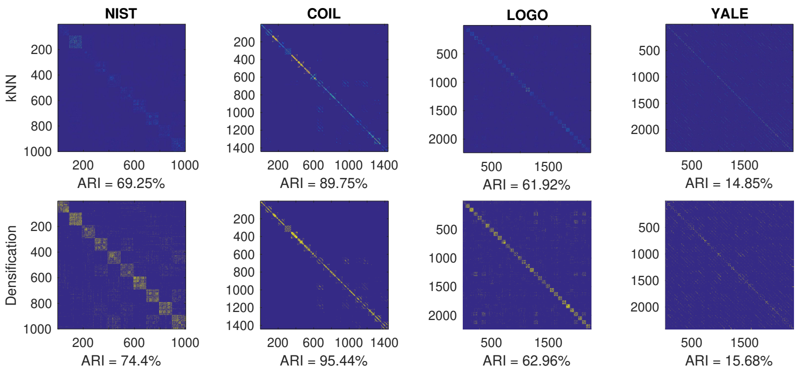


Figure 9: Best cases (Adjusted Rand Index) of densification of different datasets.

7.2. Comparison with Anchor graphs

Anchor graphs [19] are designed to produce better affinity matrices by minimizing the spectral gap λ_2. However, they require pre-computation of the

Table 1: NIST dataset: Adjusted Rand Index for different thresholds and number of k

		kNN 15			**kNN 25**			**kNN 35**		
		$\lvert E_B \rvert$								
		0.05	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*
$\lvert E'' \rvert$	*0.05*	37.3	41.88	40.62	57.23	54.33	52.26	27.12	30.88	43.49
	0.15	66.9	63.52	61.64	70.87	70.84	57.65	69.51	68.54	67.42
	0.25	71.78	69.15	65.01	71.05	70.4	70.21	69.95	71.6	70.51
	0.35	**74.4**	71.06	70.08	71.02	71.51	70.42	70.55	71.23	70.49
No dense		69.25			65.62			63.74		

optimal number of cluster representatives (model order selection). We therefore proceed to compare the performance of the parameter configuration with $|E"| = 35\%$ and $|E_B| = 5\%$ for Dirichlet densifiers with those obtained for anchor graphs with an increasing number of anchors m. In Fig. 10-left, where we explore the range $m \in [5, 900]$ for the NIST dataset. The performance of anchor graphs increases with m but degrades after reaching the maximum value at $m = 440$ (ARI = 76%). This maximum is due to the fact that anchor graphs tend to reduce the amount of inter-class noise. On the other hand, Dirichlet densifiers are completely unsupervised and do not rely on anchor computation. Their performance is constant w.r.t. m and the best of ARI obtainable is 74.4%. Our (unsupervised) densifier is only outperformed in a small range of m. These results are consistent with other datasets too. For instance, in the COIL dataset, we explore the range $m \in [5, 350]$ for the anchors. Our best ARI for the Dirichlet densifier is 95%, which is obtained with $|E"| = 35\%$ and $|E_B| = 5\%$. Anchor graphs reach a maximum ARI = 98% in the range $m \in [250, 310]$ and then degrade in performance.

Table 2: COIL dataset: Adjusted Rand Index for different thresholds and number of k

		kNN 15			**kNN 25**			**kNN 35**		
		$|E_B|$								
		0.05	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*
$|E''|$	*0.05*	55.17	57.99	33.51	54.31	51.03	30.94	72.66	71.68	67.85
	0.15	73.16	72.04	72.69	63.33	64.26	74.11	90.96	84.57	71.13
	0.25	93.69	83.68	82.98	92.09	91.09	64.32	91.01	91.99	90.27
	0.35	**95.44**	94.54	83.01	92.41	92.81	90.55	90.53	91.01	92.11
No dense		89.75			89.65			85.42		

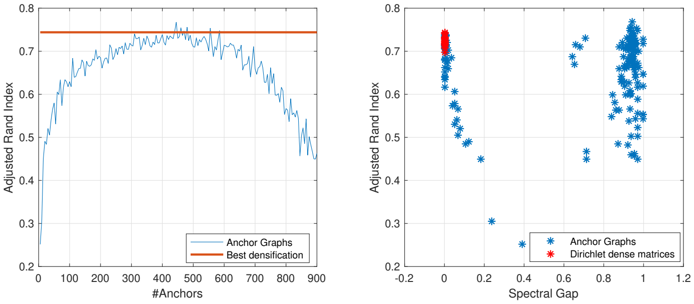


Figure 10: Left: ARIs for Anchors Graphs and Dirichlet densifiers. Dirichlet densifiers do not depend on the number of anchors and are unsupervised. Right: ARIs vs Spectral Gaps.

Finally, the good behavior of Dirichlet densifiers for minimizing λ_2 is shown in Fig. 10-right. Red dots correspond to Dirichlet dense matrices (those whose performance is reported in Table 1). This not only reconciles our results with those of the anchor graphs, but also lifts the von Luxburg and Radl's bound (Eq. 4) so that commute times can be more accurately estimated.

8. Conclusions

In principle, commute times (CTs) cannot be accurately estimated from large graphs [14]. However, in this paper we show that Dirichlet densifiers provide a

Table 3: LOGO dataset: Adjusted Rand Index for different thresholds and number of k

		kNN 15			**kNN 25**			**kNN 35**		
		$\vert E_B \vert$								
		0.05	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*
$\vert E'' \vert$	*0.05*	20.21	18.11	22.56	45.59	42.99	40.2	19.94	14.01	16.69
	0.15	60.55	58.81	56.03	57.68	47.21	48.71	52.77	14.43	51.54
	0.25	61.77	60.58	59.81	59.24	59.21	47.56	54.65	53.33	53.29
	0.35	**62.96**	61.75	61.39	60.65	59.7	58.73	57.23	55.11	53.71
No dense		61.92			59.82			54.11		

route to the computation of meaningful commute times. We highlight the fact that the spectral gap should be close to zero, and this is the role of Dirichlet densifiers (see Fig. 10-right) for small fractions of leading edges. However, the adjusted rand index (ARI) degrades linearly as the spectral gap increases. This means that the spectral gap is negatively correlated with increasing levels of inter-class noise. This noise arises when the densification level increases, since Dirichlet densifiers are still not able to confine densification to intra-class links. For anchor graphs the spectral gap is typically close to the unity. Otherwise, they outperform Dirichlet densifiers to some extent at the cost of computing anchors and finding the best number of anchors.

To conclude, we have presented a novel method for transforming graphs into denser versions which are more suitable for estimating meaningful CTs. This is due to the minimization of the Cheeger constant and, in turn, to the minimization of the spectral gap. Our method is more scalable and effective than that based on SDP. It is completely unsupervised, since our experiments with real datasets show that there are almost universal thresholds, namely $|E$”$|$ and $|E_B|$, for a variety of datasets.

Our future work includes a semi-supervised version of the Dirichlet densifier, the modification of the von Luxburg et al.’s gap to make it dependent on the

Table 4: YALE-Faces dataset: Adjusted Rand Index for different thresholds and number of k

		kNN 15			**kNN 25**			**kNN 35**		
		$\|E_B\|$								
		0.05	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*	*0.05*	*0.25*	*0.5*
$\|E''\|$	*0.05*	12.48	11.66	10.69	5.34	5.71	5.76	5.59	3.07	4.2
	0.15	15.47	14.38	13.91	10.57	10.45	9.59	7.73	7.14	6.93
	0.25	15.29	15.48	15.01	10.81	10.66	10.14	8.14	8.04	7.84
	0.35	14.46	**15.68**	15.34	9.18	6.65	10.9	7.1	7.04	7.69
No dense		14.85			9.27			7.27		

densification level and the application of densification to graph classification. We will explore also better sufficient conditions for predicting links that decrease the spectral gap. In addition, our recent work shows that densifying/rewiring the original graph has a deep implication in the improvement of graph-based ranking [30].

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, M. Tulsiani, Graph densification, in: Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012, 2012, pp. 380–392.

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

[3] F. Escolano, M. Curado, M. A. Lozano, E. R. Hancock, Dirichlet graph densifiers, in: Structural, Syntactic, and Statistical Pattern Recognition

- Joint IAPR International Workshop, S+SSPR 2016, Proceedings, 2016, pp. 185–195.

[4] F. R. K. Chung, Spectral Graph Theory, Conference Board of the Mathematical Sciences (CBMS), n 92, American Mathematical Society, 1997.

[5] P. Diaconis, D. Stroock, Geometric bounds for eigenvalues of markov chains, Ann. Appl. Probab. 1 (1) (1991) 36–61.

[6] H. Qiu, E. R., Clustering and embedding using commute times, IEEE Trans. Pattern Anal. Mach. Intell. 29 (11) (2007) 1873–1890.

[7] L. Grady, Random walks for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 28 (11) (2006) 1768–1783.

[8] S. Arora, D. Karger, M. Karpinski, Polynomial time approximation schemes for dense instances of np-hard problems, Journal of computer and system sciences 58 (1) (1999) 193–210.

[9] A. Frieze, R. Kannan, The regularity lemma and approximation schemes for dense problems, in: Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on, IEEE, 1996, pp. 12–20.

[10] J. Komlós, A. Shokoufandeh, M. Simonovits, E. Szemerédi, The regularity lemma and its applications in graph theory, in: Theoretical aspects of computer science, Springer, 2002, pp. 84–112.

[11] N. Garcia Trillos, D. Slepcev, J. von Brecht, T. Laurent, X. Bresson, Consistency of cheeger and ratio graph cuts, Journal of Machine Learning Research (17) (2016) 1–46.

[12] P. G. Doyle, J. L. Snell, Random Walks and Electric Networks, 1st Edition, Vol. 22, Mathematical Association of America, 1984.

[13] L. Lovász, Random walks on graphs: A survey, in: D. Miklós, V. T. Sós, T. Szőnyi (Eds.), Combinatorics, Paul Erdős is Eighty, Vol. 2, János Bolyai Mathematical Society, Budapest, 1996, pp. 353–398.

[14] U. von Luxburg, A. Radl, M. Hein, Hitting and commute times in large random neighborhood graphs, Journal of Machine Learning Research 15 (1) (2014) 1751–1798.

[15] U. V. Luxburg, A. Radl, M. Hein, Getting lost in space: Large sample analysis of the resistance distance, in: Advances in Neural Information Processing Systems, 2010, pp. 2622–2630.

[16] M. Alamgir, U. V. Luxburg, Phase transition in the family of p-resistances, in: Advances in Neural Information Processing Systems, 2011, pp. 379–387.

[17] C. H. Nguyen, H. Mamitsuka, New resistance distances with global information on large graphs, in: Artificial Intelligence and Statistics, 2016, pp. 639–647.

[18] Z.-Q. Luo, W.-K. Ma, A. M.-C. So, Y. Ye, S. Zhang, Semidefinite relaxation of quadratic optimization problems, IEEE Signal Processing Magazine 27 (3) (2010) 20–34.

[19] W. Liu, J. Wang, S. Kumar, 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.

[20] X. Zhu, Z. Ghahramani, J. D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International conference on Machine learning (ICML-03), 2003, pp. 912–919.

[21] J. D. Batson, D. A. Spielman, N. Srivastava, S. Teng, Spectral sparsification of graphs: theory and algorithms, Commun. ACM 56 (8) (2013) 87–94.

[22] K. C. Toh, M. J. Todd, R. H. Tutuncu, SDPT3-a Matlab software package for semidefinite programming, version 1.3, Optimization Methods and Software 11 (1-4) (1999) 545–581.

[23] S. O. Haikin, Neural Networks and Learning Marchines (3rd Edition), Prentice-Hall, 2009.

[24] M. Fiedler, A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory, Czechoslovak Mathematical Journal 25 (100) (1975) 619–633.

[25] R. Eldan, M. Z. Rácz, T. Schramm, Braess's paradox for the spectral gap in random graphs and delocalization of eigenvectors, Random Struct. Algorithms 50 (4) (2017) 584–611.

[26] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.

[27] S. A. Nene, S. K. Nayar, H. Murase, Columbia object image library (coil-20, Tech. rep. (1996).

[28] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep. (2009).

[29] D. Cai, X. He, J. Han, Spectral regression for efficient regularized subspace learning, in: Proc. Int. Conf. Computer Vision (ICCV'07), 2007.

[30] M. Curado, F. Escolano, M. Lozano, E. Hancock, Net4lap: Neural laplacian regularization for ranking and re-ranking, 2018, 24th International Conference on Pattern Recognition ; Conference date: 21-08-2018 Through 24-08-2018.

***Author Biography**

Manuel Curado received his degrees in technical engineering and computer science from the University of Cordoba, Spain, in 2009 and 2013, respectively, and his PhD in Computer Science from the University of Alicante (2018). He is currently working as Associate Professor in the Catholic University of Avila. He has conducted research stays at the Computer Vision and Pattern Recognition Lab of the University of York (2017), as well as the European Centre for Living Technology lab at the University Ca'Foscari of Venice (2016), and at the University of Barcelona (2018). His research interests include pattern recognition, machine learning, and computer vision.

Francisco Escolano received his B.S. degree in computer science from the Polytechnical University of Valencia, Spain, in 1992 and his Ph.D. degree in computer science from the University of Alicante in 1997. Since 1998 he has been an associate professor with the Department of Computer Science and Artificial Intelligence of the University of Alicante. He has been a postdoctoral fellow with Dr. Norberto M. Grzywacz at the Biomedical Engineering Department of the University of Southern California, Los Angeles, CA, and he has also collaborated with Dr. Alan Yuille at the Smith-Kettlewell Eye Research Institute, San Francisco, CA. He also visited the Liisa Holm's Bioinformatics Lab at the University of Helsinki. He was both co-funder and head of the Robot Vision Group of the University Alicante from 2001 to 2012. He is now a member of University of Alicante's Mobile Vision Research Laboratory (MVRLab), He has chaired the TC2 (Structural Pattern Recognition) committee of the IAPR from 2012 to 2016. He now serves as vice-chair of this committee.

Miguel Angel Lozano received his degree (2001) and a PhD (2008) in Computer Engineering from the University of Alicante. Since 2004 he has been a lecturer at the Department of Computer Science and Artificial Intelligence of the University of Alicante. He has conducted research stays at the Computer Vision and Pattern Recognition Lab of the University of York, as well as the Bioinformatics Lab at the University of Helsinki. He is currently the head of the University of Alicante's Mobile Vision Research Laboratory (MVRLab), the director of the Master's Degree in Software Development for Mobile Devices, and the Coordinator for Quality Assurance and Educational Innovation at the Polytechnic School. His research interests include pattern recognition and computer vision.

Edwin Robert Hancock holds a BSc degree in physics (1977), a PhD degree in high-energy physics (1981) and a D.Sc. degree (2008) from the University of Durham, and a doctorate Honoris Causa from the University of Alicante in 2015. He is currently Emeritus Professor in the Department of Computer Science at the University of York, and also Adjunct Professor and Principal Investigator - Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University. He became a fellow of the International Association for Pattern Recognition in 2000, the Institute of Physics in 2007 and the IEEE in 2016. He is currently Editor-in-Chief of the journal Pattern Recognition, and was founding Editor-in-Chief of IET Computer Vision from 2006 until 2012. He has also been a member of the editorial boards of the journals IEEE Transactions on Pattern Analysis and Machine Intelligence, Pattern Recognition, Computer Vision and Image Understanding, Image and Vision Computing, and the International Journal of Complex Networks. He was Conference Chair in 1994 and Programme Chair in 2016 for the British Machine Vision Conference, Track Chair for ICPR (2004 and 2016) and Area Chair for ECCV (2006) and CVPR (2008 and 2014). In 1997 he jointly established the EMMCVPR workshop series with Marcello Pelillo. He was awarded a Royal Society Wolfson Research Merit Award in 2009, was named BMVA Distinguished Fellow in 2016 and received the IAPR Pierre Devijver Award in 2018. He was a Governing Board Member of the IAPR (2006-2016), and was Second Vice President of the Association (2016-2018). He was a member of the Computer Science and Information Technology sub-panel for the UK REF 2014 and will also be a panelist for REF 2021. He has published about 200 journal papers and 650 refereed conference papers.