Network Filtering for Big Data: Triangulated Maximally Filtered Graph

Guido Previde Massara[1], T. Di Matteo[1,2], Tomaso Aste[1,3]
[1] *Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK*
[2] *Department of Mathematics, King's College London, The Strand, London, WC2R 2LS, UK*
[3] *Systemic Risk Centre, London School of Economics and Political Sciences, London, WC2A2AE, UK*
(Dated: 16/07/15)

We propose a network-filtering method, the Triangulated Maximally Filtered Graph (TMFG), that provides an approximate solution to the Weighted Maximal Planar Graph problem. The underlying idea of TMFG consists in building a triangulation that maximizes a score function associated with the amount of information retained by the network. TMFG uses as weights any arbitrary similarity measure to arrange data into a meaningful network structure that can be used for clustering, community detection and modeling. The method is fast, adaptable and scalable to very large datasets, it allows online updating and learning as new data can be inserted and deleted with combinations of local and non-local moves. TMFG permits readjustments of the network in consequence of changes in the strength of the similarity measure. The method is based on local topological moves and can therefore take advantage of parallel and GPUs computing. We discuss how this network-filtering method can be used intuitively and efficiently for big data studies and its significance from an information-theoretic perspective.

Keywords: TMFG, Big Data, Network Filtering, PMFG, Planarization algorithms, Correlation Network, Markov Random Fields, Weighted Maximal Planar Graph (WMPG)

I. INTRODUCTION

We are witnessing interesting times rich of information, readily available for us all. Using, understanding and filtering such information has become a major activity across science, industry and society at large. Our society has become a global information processing system where news propagate and impact on individuals and the economy at increasingly fast rates with increasingly large effects. It is therefore important to have tools that can analyse this information while it is generated and that can provide ways to reduce complexity and dimensionality while keeping the integrity of the dataset. Information content and flow are often associated with large degrees of redundancy both in time (repeating and scaling patterns) and across different variables (similarity, dependency and causality). Redundancy is often used to convey strength to the meaning or, more simply, it is the signal of recurring patterns with high statistical significance and therefore important. In this paper we propose to use such redundancy to build an information-based network that retains the relevant part of the data-interdependency structure. The structure of this network is a representation of the information in the dataset and such information can be efficiently analysed by using network-theoretic tools.

The idea of using redundancy – mostly correlation coefficients – to filter information in complex datasets by building sparse networks retaining relevant edges only has been very actively studied in the literature mostly by means of two approaches: i) the minimum spanning tree (MST) [1, 2]; ii) the planar maximally filtered graph (PMFG) [3–5]. The common idea underneath these two approaches is to filter a dense matrix of weights by retaining the largest and most significant possible subgraph while imposing global constraints on the topology of the resulting network. In particular, in the MST approach edges with the largest weights (e.g. correlations) are retained while constraining the subgraph to be globally a (spanning) tree. Similarly, in the PMFG construction the largest weights (e.g. largest correlation coefficients) are retained while constraining the subgraph to be globally a planar graph (see [2–4]). Both the MST and

the PMFG are particular cases of simplicial complexes; our proposed method can be extended to more general simplicial complexes with different constraints (for instance on the topological genus, or on the size of the largest complete subgraph – or *clique*).

The PMFG is a greedy solution of the WEIGHTED MAXIMUM PLANAR GRAPH (WMPG) problem: given a complete edge-weighted graph find a planar subgraph which is maximal (i.e. no edge can be added without destroying planarity) and such that the sum of the edge weights is maximum. The problem is known to be NP-complete (see [6] for a proof and [7] for a review), but algorithms providing sub-optimal solutions are known. For instance, an approximation algorithm with a guaranteed performance ratio of $\frac{3}{8}$ for complete graphs is discussed in [8].

The PMFG has a richer information content than the MST with a larger number of edges (the PMFG has $3p-6$ edges, while the MST has $p-1$, where p is the number of vertices) and contains of 3- and 4-cliques. However, the network is still sparse, filtering $3p-6$ edges out of $p(p-1)/2$ of the complete graph K_p which is associated with the original dense matrix of weights.

Planar filtered graphs are powerful tools to study complex datasets. It has been shown in [9] that by making use of the 3-clique structure of the PMFG a clustering structure can be extracted allowing dimensionality reduction that keeps both local information and global hierarchy in a deterministic manner without the use of any prior information. Applications of Planar filtered graphs to financial data-sets can meaningfully identify industrial activities and structural market changes [10, 11]. Planar filtered graphs can be used to diversify financial risk by building a well-diversified portfolio that effectively reduces risk by investing in stocks that occupy peripheral regions of the graph [12]. Planarity ensures easy visualization of the network with the possibility to draw the network without edge-crossing. Another appealing advantage of planar filtered networks concerns graphical modeling (e.g. Markov Random Fields [13]) where planarity (which limits the treewidth of the junction tree of the filtered graph) grants that some exact inference algorithms can be performed in an efficient fashion (see [14, 15]). However, the algorithm so far proposed to construct the PMFG is computationally costly and cannot be applied to large datasets. There is therefore scope to search for novel algorithms that can construct planar filtered graphs in a computationally efficient way. In the present paper we indeed introduce a computationally efficient algorithm, the TMFG, that produces planar filtered graphs by optimizing an objective function (which we shall call "score function") by using local topological moves called T_1 and T_2 [16], the 'Alexander move' A [17], in conjunction with a 'vertex-swap' operator S. These topological moves are sketched in Figs. 1, 3, 5 and 6. The T_1 move acts on two triangles which share an edge and replaces the shared edge with a new edge joining the (previously) opposite vertices (see Fig. 3); the T_2 move adds a vertex inside a triangle and connects this vertex to the triangle's vertices with three new edges (see Fig. 1); the 'A move operates on two triangles which share an edge by deleting the shared edge, adding a new vertex inside the resulting rhombus and joining the added vertex to the rhombus' vertices (see Fig. 5); finally the S operator swaps two vertices of the graph keeping fixed the neighbours (see Fig. 6).

The TMFG algorithm has also the advantage of allowing 'online' updates of the planar graphs. Furthermore, the TMFG can be naturally applied to multipoint dependency measures associated with the 3- and 4-clique structure. When only T_2 and S operators are used the algorithm produces triangulated (or chordal) graphs together with their structure of cliques and separators. Chordal graphs have several appealing properties being very well suited to modelling with Markov Random Fields (MRF) with a computationally attractive closed-form solution for the Maximum Likelihood Estimate of the joint probability (see [18, 19]), they are also *perfect graphs* and as such have polynomial time solutions for problems that are otherwise harder (e.g. graph coloring problem, maximum clique problem, and maximum independent set problem, see [20]). ~~Graphical models with a chordal underlying graph have a particularly useful representation of the joint probability distribution and a closed form solution for the Maximum Likelihood Estimate (MLE).~~ In the case of Gaussian graphical models the structure of the graph represents partial correlations between variables and the non-zero entries of the *concentration* matrix (the inverse

of the covariance matrix also called *precision* matrix [18, 21, 22]) coincide with the edges of the graph. Additionally, the algorithm has the advantage that it is not restricted to planar topologies allowing higher-genus hyperbolic embeddings to be explored [16, 23–25]. Finally, given its local nature, the algorithm is ideally suited for parallelisation and GPU computing.

This paper is organized as follows. In section II we discuss some facts about Planar Maximally Filtered Graphs describing two algorithms used to generate such graphs; in section III we introduce the TMFG algorithm and we highlight some characteristics of the algorithm that are relevant to applications with high-dimensional, frequently-updated datasets; in section IV we offer some information-theoretic perspectives on the selection of a particular score function; finally in section V we apply TMFG to several weight distributions showing that it is computationally faster than PMFG achieving comparable or better results.

II. PLANAR MAXIMALLY FILTERED GRAPHS

Algorithms for the extraction of planar subgraphs from dense networks are relevant in several domains such as, for instance: i) the analysis of financial data – where nodes generally represent financial assets (such as stock prices, spreads, liabilities, risk or liquidity indicators etc.) and the edges represent correlations or other measures of dependence between them (see [3, 4, 12, 26]); ii) facilities layout – where nodes represent the facilities and the edges the affinities between them (see [27–29] for a survey); iii) integrated circuit design – where nodes are the electrical elements and connections are the physical connections (see [30]); iv) systems biology – where nodes can represent proteins and edges protein interactions in a metabolic network (see [31]); v) social systems – where nodes represent social agents (e.g. individuals, companies, groups) and edges represent social interaction (see [32] for a detailed overview). In some domains – such as facility layout or integrated circuit design – the constraint of planarity is a direct consequence of the two dimensional planar geometry of the problem, while in other domains, network planarity is a way to constraint the complexity of the graph reducing the degree of interwovenness [3]. Planarity is also desirable because many NP-hard problems have efficient polynomial-time solutions, or better approximations, for planar graphs (vertex coloring, edge coloring, independent vertex set, multicommodity flows, see [33] for an introduction).

Let us here briefly review two known algorithms that have been used to construct planar filtered networks from a matrix of weights:

- PMFG [4];
- Deltahedron heuristics and subsequent improvement [7, 28, 29].

These algorithms provide estimates for optimal solutions to the Maximum Weighted Planar Graph problem. Let us here recall that, given a complete edge-weighted graph $G(V, E)$, with vertex set V and edge set E, the Maximum Weighted Planar Graph problem requires to build a planar subgraph $G'(V, E')$, with $E' \subset E$ such that adding another edge $e \in E \setminus E'$ would cause $G(V, E' \cup e)$ to be non planar and such that the sum of the weights is maximum (see [29] and [7] for a detailed description of the problems and a survey).

A. Planar Maximally Filtered Graph

The PMFG algorithm [3] searches for the maximum weighted planar subgraph by adding edges one by one (see [5]). The resulting matrix is sparse, with $3(p-2)$ edges. The algorithm starts by sorting by weighting all the edges of a dense matrix of weights in non increasing order and tries to insert every edge in the PMFG in that order. Edges that violate the planarity constraint are

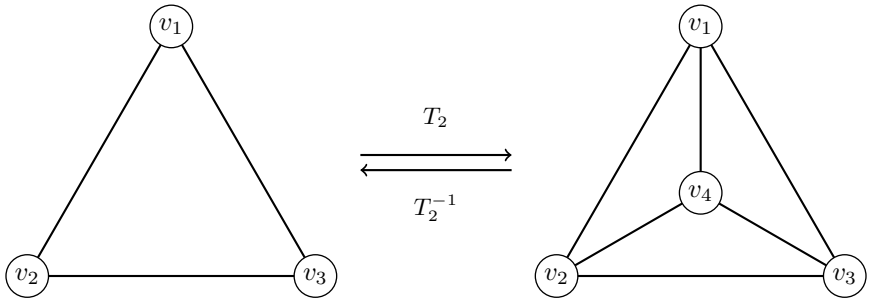

FIG. 1: T_2 move: addition of one vertex within a triangular face [16, 23, 24, 38, 39]. Its inverse, T_2^{-1}, removes a vertex from inside a three-clique (in this case the clique $\{v_1, v_2, v_3\}$).

discarded. The most computationally intense part of the algorithm is the planarity test, which is performed every time an edge insertion is attempted. It results that the PMFG construction performs an order of p^2 ($O(p^2)$) planarity tests on any dense $p \times p$ matrix of weights W. Assuming that the complexity for a planarity test in $O(p)$ (see [34, 35]) the computational complexity of the whole algorithm results in a $O(p^3)$ [9].

B. Deltahedron heuristic

The deltahedron heuristic [28, 29] searches for approximate solutions of the WMPG problem starting from a tetrahedron, K_4, which is planar. Then, at each step a vertex is added into a triangular face and three edges are added connecting the newly inserted vertex to the vertices of the triangular face. This vertex insertion in a triangular face is called T_2 move (see Fig.1 and [16, 23, 24, 38, 39]). It is easy to see that the T_2 operator acts without breaking planarity ensuring that the final network is planar. The triangular face is chosen in order to maximise the sum of the newly connected edges, while the vertices to be inserted are extracted from a pre-sorted list. The vertex list can be sorted according to two functions of the edge weights incident to the vertex, yielding two possible variants of the deltahedron heuristic:(i) the sum of the incident edge-weights or (ii) the maximum incident edge-weight. Different weightings lead to different ordering for the vertices and different results.

The deltahedron heuristic algorithm is not "greedy" (unlike the PMFG that chooses the heaviest feasible edge at every step) since the choice of the ordering of the vertices is done once at the beginning and there is no subsequent attempt at optimising the order of the vertices taking into account the local configuration. ~~and there is no known performance guarantee.~~ However the algorithm is considerably faster than the PMFG, since every T_2 move keeps the planarity of the graph and therefore there is no need to test for planarity at each stage.

An important feature of the graphs produced by T_2 moves is that they are chordal graphs: every cycle of length greater than 4 has a *chord*, an edge not belonging to the cycle that joins two non-adjacent vertices. Chordal graphs are *perfect* graphs and as such there are polynomial time algorithms for solving generally hard problems such as finding a maximum clique, graph coloring, and maximum independent set.

In [7, 40] the deltahedron heuristic is improved by maintaining a record of the most favourable vertex insertion moves (Green and Al-Hakim heuristic – the GH-heuristic henceforth), essentially keeping a cache of the best and next-to-best options for inserting any of the remaining vertices. The cache is updated as the algorithm progresses. Optionally Osman et al. [7] allow for a

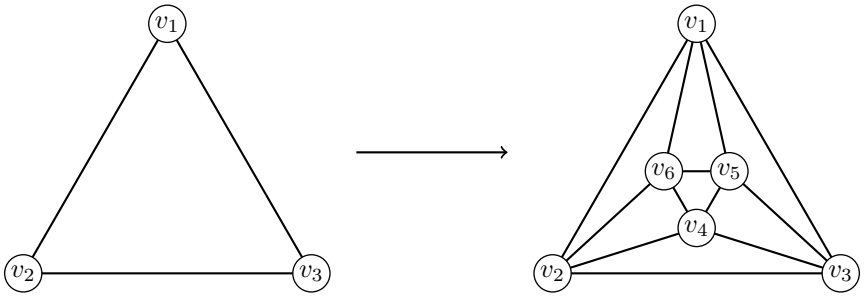

FIG. 2: Addition of three vertices in Leung's extension of the deltahedron heuristic.

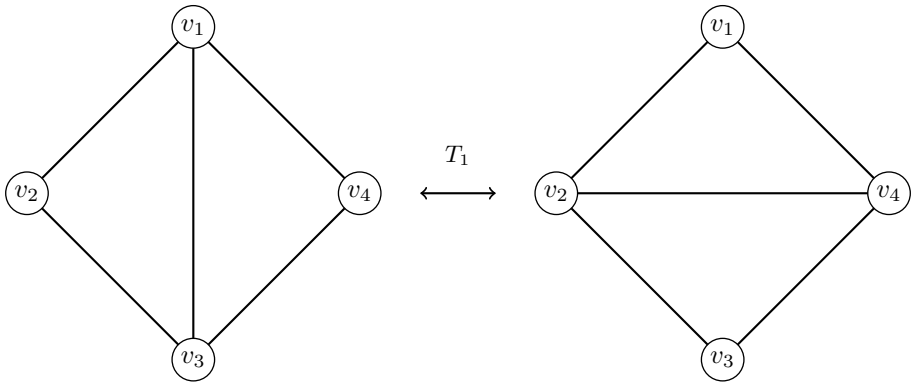

FIG. 3: T_1 move: rewiring of a shared edge between neighboring triangular faces.

parameter that governs the"greediness" of the algorithm. In section III we will introduce a modified version of the GH-heuristic algorithm.

C. Local topological moves: $\mathbf{T_1}$, $\mathbf{T_2}$, **A**, & **S**

With the deltahedron heuristic we have already introduced the T_2 move that, as shown in Fig.1, inserts vertex v_4 into the triangular face $\{v_1, v_2, v_3\}$ splitting it into three triangular faces $\{v_1, v_2, v_4\}$, $\{v_1, v_4, v_3\}$, and $\{v_4, v_2, v_3\}$. In the following we will call *face* a three-clique that does not contain any vertex in the given embedding, reserving the word *triangle* for a generic 3-clique. We see that, after the T_2 move, $\{v_1, v_2, v_3\}$ is no longer a face but rather a 3-clique.

In an extension of the deltahedron heuristic method, suggested by Leung [41], vertex insertion can happen either one vertex at a time (the T_2 move, as in Fig.1) or three vertices at a time as in Fig.2. This corresponds to the insertion of an octahedron within a triangular face [25]; clearly this is different from the T_2 move that instead corresponds to the insertion of a tetrahedron. However,

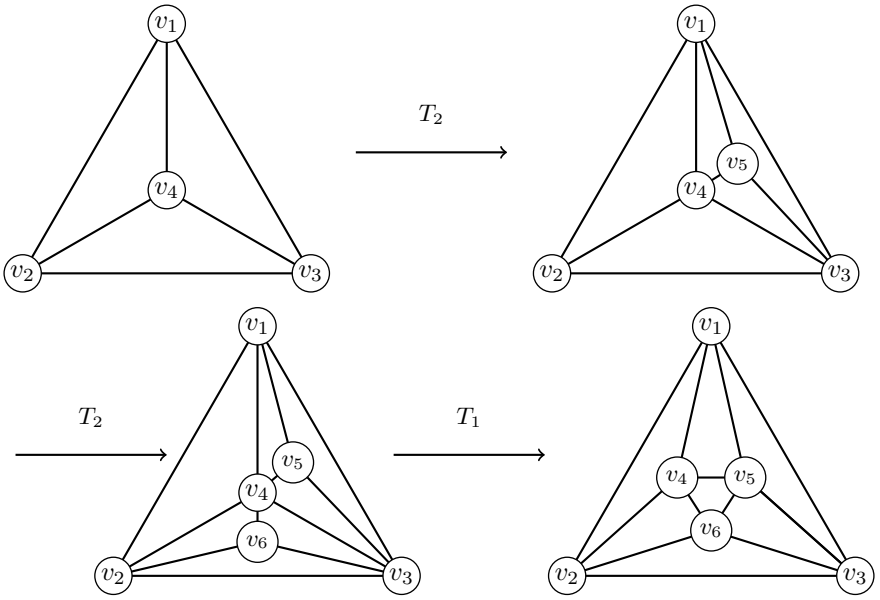

FIG. 4: Demonstration that the Leung's extension in Fig.2 can be generated by using two T_2 and one T_1 moves.

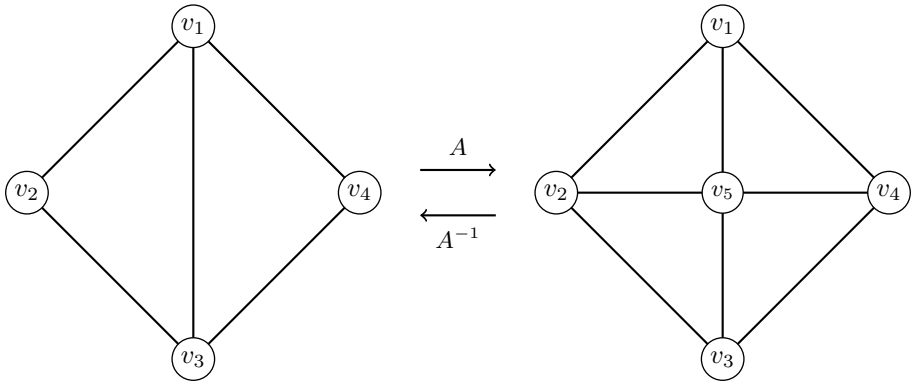

FIG. 5: A move: insertion of a vertex inside a plaquette made of two neighbouring triangular faces.

such a move can be obtained by combining T_2 with another local move, called T_1 [16, 23, 24, 38], consisting in switching neighbors among two adjacent triangles, as shown in Fig.3. In general, any local topological change of a surface triangulation that preserves embedding and results in a triangulation can be realized through the combination of the two elementary moves T_1 and T_2 [17]. However it should be pointed out that the application of T_1 could cause the graph to become no longer chordal. For instance, the Leung extension (Fig.2) can be produced via two T_2 and one T_1, as demonstrated in Fig.4.

Another move that we will use to build planar graphs is the A move as described in Fig.5. Also in this case the move can be produced combining T_1 and T_2 and leads to non-chordal graphs.

Finally, we will use the 'swap' operator, S, that re-labels sub sets of vertices of a graph as shown in Fig.6, where it is acting on the vertices of a 4-simplex. This operation is trivial when the weights are identical, but will in general affect aggregate functions of the weights in a non-trivial

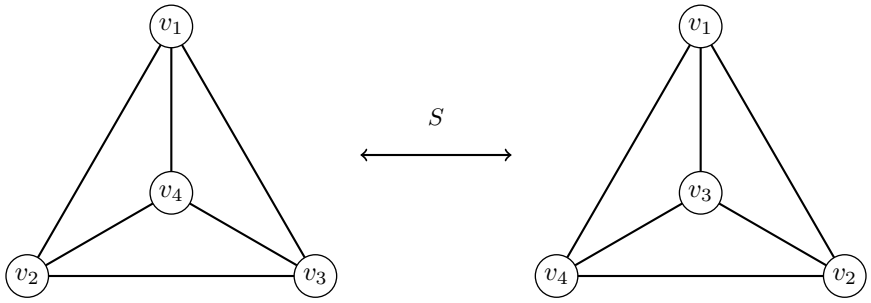

FIG. 6: S move: relabelling of the vertices of a 4-simplex. Note that the topology of the graph is unchanged.

way. The peculiarity of this operator is that it has not to operate locally and it keeps topology unchanged preserving therefore planarity.

In the following section we shall see how T_2, T_1, A and S moves can be used to generate planar filtered graphs as well as higher genus, non-planar, filtered graphs.

III. TRIANGULATED MAXIMALLY FILTERED GRAPH

A. TMFG construction

The TMFG algorithm starts from a clique of order 4 (K_4) and adds vertices by using the local move T_2. The novelty is that, at each step, the algorithm optimizes a *score function* (e.g. the sum of the weights of the edges). Similarly to the GH-heuristics, the method does not rely on any particular ordering of the vertices but, at every step, it calculates the score that would be obtained by adding any of the remaining vertices inside any feasible face. T_2 is applied to the vertex and face pair that leads to the maximum increase in score. A naive implementation would require to evaluate the gain function for every pair consisting of a feasible vertex and a feasible face, thus resulting in an $O(p^2)$ calculations at every step and therefore $O(p^3)$ overall computational complexity. However, it is possible to maintain and update incrementally a cache with the information about the best possible pairing updating only the records affected by a move. This cache contains as many elements as there are feasible faces ($O(p)$). Since the calculation of the maximum of a vector of $O(p)$ elements requires $O(p)$ calculations , the overall number of calculations for the score functions is $O(p^2)$. This results in much faster computational times with respect to the $PMFG$. Differently from [7] we use a slightly different data structure to keep track of the vertices to insert into the feasible faces. We also keep track of the triangles that are no longer faces because this is relevant for subsequent modelling (see section IV).

Let us first focus on T_2 moves only. After every application of T_2 the cache is updated: some scores that were previously achievable are no longer feasible, while others become feasible and the corresponding score is calculated. More formally, we define a *score function* $S(v_h, \{v_a, v_b, v_c\})$ that quantifies the gain achievable by adding vertex v_h inside the triangle $\{v_a, v_b, v_c\}$.

For instance, for a given, dense, matrix of weights W, the gain function can be the sum of the weights of the edges that will be added by inserting v_h in face $\{v_a, v_b, v_c\}$: $S(v_h, \{v_a, v_b, v_c\}) = W(v_h, v_a) + W(v_h, v_b) + W(v_h, v_c)$. In the next session we discuss an information theoretic interpretation of the score function.

The cache is a structure made up of two vectors ($MaxGain$ and $BestVertex$) indexed by the faces in the planar graph present up to that point. Let us consider a given stage of the construction with m triangular faces t_i, $i \in \{1, 2, \cdots, m\}$ and k remaining uninserted vertices $v \in \{v_1 \cdots v_k\}$. The $MaxGain$ vector contains the value of the maximum gain over all remaining vertices for all triangular faces:

$$\mathbf{MaxGain} = \left(\max_{v \in \{v_1 \cdots v_k\}} S(v, t_1), \max_{v \in \{v_1 \cdots v_k\}} S(v, t_2), \ldots \max_{v \in \{v_1 \cdots v_k\}} S(v, t_m) \right) \quad . \tag{1}$$

The $BestVertex$ vector contains inside the list of vertices that attains the maximum gain for the specific triangular face:

$$\mathbf{BestVertex} = \left(\arg\max_{v \in \{v_1 \cdots v_k\}} S(v, t_1), \arg\max_{v \in \{v_1 \cdots v_k\}} S(v, t_2), \ldots, \arg\max_{v \in \{v_1 \cdots v_k\}} S(v, t_m) \right) \tag{2}$$

When a vertex (say vertex v_h) is added to a certain triangular face (say face t_j) the two cache vectors must be updated by removing vertex v_h from the list of remaining vertices, removing face t_j and adding three new faces. It is worth noting that t_j becomes a *clique separator* of the graph [22]. The TMFG pseudocode is shown in Algorithm 1. For simplicity we have not given details of the application of the moves T_1, A and the swap operator S. The TMFG generated by using T_2 only is a 4-clique tree.

The TMFG algorithm can be extended to include T_1 and A moves as well. In this case, the moves are local, internal to the plaquette made by two joint triangles (i.e. $\{v_1, v_2, v_3\}$ and $\{v_2, v_3, v_4\}$ in Fig.3). The gain function for a T_1 move is associated to the removal of an edge (i.e. (v_1, v_3) in Fig.3) and the simultaneous addition of another edge (i.e. (v_2, v_4) in Fig.3). Similarly the gain for a move of type A (as shown in Fig.5) results from the removal of one edge and the insertion of a new vertex and four new edges. The use of T_1 and A moves generally improve gain; however, we have verified that the algorithm with T_2 only produces very similar results. Furthermore, planar filtered graph with T_1 or A moves are no longer clique trees but rather bubble-trees [9] which are in general no longer chordal. For instance see Fig.4 where the application of T_1 creates a non-chordal graph: the cycle $v_1 - v_2 - v_6 - v_5 - v_1$ has length grater than 3 without internal chords. This can have some implications for dependency modeling, as we shall discuss in Section IV. In the following we will therefore consider separately the two cases of TMFG constructed with and without T_1 and A. In many cases the application of the swap operator S results in higher overall gains. This operator has the advantage of leaving the overall topology unchanged but its use should be regulated by few local or heuristic criteria to avoid an increase in the complexity of the algorithm due to the increasing number of possible combinations. The case that we have implemented requires the evaluation of all the possible combinations of the four vertices involved in the execution of a T_2 operation. This requires some further changes to the cache vectors, but – being applied locally – it does not increase the overall computational complexity that remains $O(p^2)$.

The TMFG algorithm is not greedy with respect to edge insertion in the sense that the best possible move is chosen from a *subset* of all the feasible edge insertions that preserve planarity. Nonetheless, we shall see that TMFG performs as well as – or better than – the PMFG for a large class of weight matrices, including squared correlation coefficient matrices from empirical time series which are relevant for modeling [42].

input : A dense $p \times p$ square matrix $\mathcal{W}$ with positive weights (e.g. a matrix of squared correlation coefficients)
output: A sparse matrix, $\mathcal{P}$, a filtered version of $\mathcal{W}$ fulfilling the planarity constraint

$\mathcal{C}_1 \leftarrow$ Tetrahedron, $\{v_1, v_2, v_3, v_4\}$, with highest overall total score ;
// Assign the four triangular faces in $\mathcal{C}_1$ to the array $\mathcal{T}$
$\mathcal{T} \leftarrow \{\{v_1, v_2, v_3\}, \{v_1, v_2, v_4\}, \{v_1, v_3, v_4\}, \{v_2, v_3, v_4\}\}$;
// Put the $p-4$ vertices not belonging to $\mathcal{C}_1$ in the array $\mathcal{V}$
$\mathcal{V} \leftarrow \{v_5, \cdots, v_p\}$;
// Create an empty list of Separators
$\mathcal{S} \leftarrow \varnothing$;
// Assign the first tetrahedron to the list of cliques
$\mathcal{C} \leftarrow \mathcal{C}_1$;
$\mathcal{P} \leftarrow \mathcal{W}(\mathcal{C}_1, \mathcal{C}_1)$;
Calculate $MaxGain$ for $\mathcal{T}$ and $\mathcal{V}$ as in Eq.(1) ;
Calculate $BestVertex$ for $\mathcal{T}$ and $\mathcal{V}$ as in Eq. (2) ;
// Insert $p-4$ vertices via T_2
while $\mathcal{V}$ *is not empty* **do**
  Find the $t_i \in \mathcal{T}$ and $v_i \in \mathcal{V}$ that achieve the maximum in $MaxGain$;
  Insert v_i into t_i // this creates three new triangles t_a, t_b, t_c
  $\mathcal{V} \leftarrow \mathcal{V} \setminus v_i$;
  $\mathcal{T} \leftarrow (\mathcal{T} \setminus \{t_i\}) \cup \{t_a, t_b, t_c\}$;
  $\mathcal{S}_i \leftarrow \{t_i\}$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_i$;
  $\mathcal{C}_i \leftarrow \{t_i, t_a, t_b, t_c\}$;
  $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_i$;
  $\mathcal{P} \leftarrow \mathcal{P} + \mathcal{W}(\mathcal{C}_i, \mathcal{C}_i) - \mathcal{W}(\mathcal{S}_i, \mathcal{S}_i)$;
  Update $MaxGain$ and $BestVertex$ to reflect the changes in $\mathcal{T}$ and $\mathcal{V}$;
end
return $\mathcal{P}$;

Algorithm 1: TMFG algorithm

B. Dynamical adaptability

Due to the local nature of the operators, T_1, T_2, T_2^{-1}, A, A^{-1} and (local) S, used to construct the TMFG one can continuously modify the network allowing 'online' adaptability while new data are generated. This is of practical importance because in real, big data, applications information is changing dynamically with new data continuously fed causing changes in the matrix of weights that require modifications of the filtered graph. Further, creation of new nodes is required when new elements/variables become relevant in the system. Conversely, elements/variables can eventually become irrelevant and the corresponding vertices should be eliminated from the graph. The implementation of these moves requires keeping a cache matrix of gains continuously updated and dynamically checking for moves that improve total gains.

C. Parallelization and big data

The local nature of TMFG construction and dynamical adaptation through T_1, T_2, T_2^{-1}, A, A^{-1} and (local) S, moves make it ideal for parallelization. There are several possibilities for parallelization and it is beyond the purpose of the present work to implement a parallel algorithm for TMFG. Let us however discuss briefly a possible parallel implementation of the TMFG. One of the main features of planar triangulations is that three-cliques uniquely divide the network into two 'inside' and 'outside' subgraphs within a nested hierarchical structure [43]. This means that,

given a seed structure of three-cliques, each clique can develop its inside subgraph independently. A processor can be assigned to each seed clique and calculations can be performed locally. Given that each separating clique divides roughly the graph into two parts one can compute the TMFG in $O(p)$ using $O(\log p)$ processors. Another issue related to big data is the size of the score vectors. It is clear from the construction that the size of the cache grows linearly with the dimension of the problem and that triangles in the basis can be assigned to different processors, allowing parallel updates of the cache.

D. Memory usage

In the case of pair-wise dependence (such as correlation) both the deltahedron heuristic and the PMFG require to compute in advance the entire correlation matrix, while the TMFG does not use the full information from the correlation matrix and could calculate only the correlations necessary for the incremental update of the gain vectors. This is an advantage already for correlation measures, in the (numerous) cases where the number of observations (q) is less than the number of variables (p): in fact it could require much less memory (approximately $p \times q$) to store the time series of the observations and calculate the correlations on-demand, rather than calculating and storing a large correlation matrix (approximately $\frac{p\times(p-1)}{2}$). This fact is even more relevant for multi-point dependencies (e.g. partial correlation, mutual information, ...): in these cases the TMFG would still require only to store the time series in memory and would require the calculation of the relevant gain functions only, while other methods would require the storage of large amounts of data (e.g. order of p^3 for a three-points dependency measure).

IV. MODELING WITH TMFG: INFORMATION THEORETIC PERSPECTIVE

In complex systems, such as financial markets, a large number of interdependent variables are typically involved. The TMFG is a way of filtering the structure of interrelation between the variables reducing it to a network of most relevant interactions.

Modeling the system statistically consists in identifying the joint probability distribution that best describes the observed collective behavior of the variables.

Specifically, given a set of observations $\{x_1(1), ..., x_1(q)\}$, $\{x_2(1), ..., x_2(q)\}$, ... $\{x_p(1), ..., x_p(q)\}$ of p random variables $\mathbf{X} = \{X_1, X_2, ..., X_p\}$, one aims to estimate a joint probability distribution function $Q(\mathbf{X})$ that is the best representation of the 'true' multivariate probability distribution function $P(\mathbf{X})$ from which the set of observations are drawn. Clearly, $P(\mathbf{X})$ is unknown and the only information available are the observations $\{x_1(1), ..., x_1(q)\}$, $\{x_2(1), ..., x_2(q)\}$, ... $\{x_p(1), ..., x_p(q)\}$ from which $Q(\mathbf{X})$ must be estimated.

Information filtering graphs can be used to compute $Q(\mathbf{X})$. The main advantage is that these graphs are locally low dimensional (e.g. the largest clique is K_4 when planarity is enforced) which makes sampling tractable also with limited amount of data [42]. Further, the TMFG constructed from T_2 moves results in a tree made of 4-cliques separated by 3-cliques (called separators in the following). This can be seen by observing that the root of the tree is the initial 4-clique and that every T_2 move creates a further 4-clique one level deeper, and that every 4-clique other than the root is separated from its parent by a 3-clique. A pictorial representation of this fact is in the first two pictures in Fig. 2: the T_2 move creates the clique $\{v_1, v_3, v_4, v_5\}$ one level underneath the parent clique $\{v_1, v_2, v_3, v_4\}$ and the two cliques are separated by the 3-clique $\{v_1, v_3, v_4\}$. This is a particular case of a *triangulated* or *chordal* or *decomposable* graph [22]. From the theory of graphical models (see [18]) we know that the probability distribution function associated to a

decomposable graph – such as the TMFG – admits the representation:

$$Q(\mathbf{X}) = \frac{\prod_{c\in\mathcal{C}} P_c(\mathbf{X}_c)}{\prod_{s\in\mathcal{S}} P_s(\mathbf{X}_s)} \ . \tag{3}$$

Where: $\mathcal{C}$ and $\mathcal{S}$ are respectivelly the set of cliques and separators of the graph and $P_c(\mathbf{X}_c)$ and $P_s(\mathbf{X}_s)$ are the marginal probabilities of the sub sets of variables X_c and X_s associated respectively with the 4-clique c and the triangular separator s.

Equation (3) reduces the p-dimensional problem of estimating the joint probability distribution function $Q(\mathbf{X})$ to the estimation of a set of 3- and 4-dimensional local marginal probabilities $P_s(\mathbf{X}_s)$ and $P_c(\mathbf{X}_c)$ [42]. Such a reduction of a global high-dimensional problem to a set of local low-dimensional problems helps greatly in the estimation of the joint probability. The open question is now to measure how well the, unknown, true joint distribution $P(\mathbf{X})$ is represented by the model estimation $Q(\mathbf{X})$ factorized over the TMFG. To this end we can measure the dissimilarity between the two probability density functions which is given by the Kullback-Leibler divergence [44]:

$$D_{KL}(P \parallel Q) = \sum_{\mathbf{X}} P(\mathbf{X}) \log\left(\frac{P(\mathbf{X})}{Q(\mathbf{X})}\right) \ ; \tag{4}$$

For simplicity we are considering discrete variables, a similar treatment can be developed for continuous variables. The goal is to construct the TMFG that minimizes such a distance. By substituting Eq.3 into Eq.4 we write the Kullback-Leibler divergence as follows:

$$\begin{aligned} D_{KL}(P \parallel Q) = &\sum_{\mathbf{X}} P(\mathbf{X}) \log\left(P(\mathbf{X})\right) \\ &- \sum_{c\in\mathcal{C}} \sum_{\mathbf{X}_c} P_c(\mathbf{X}_c) \log\left(P_c(\mathbf{X}_c)\right) \\ &+ \sum_{s\in\mathcal{S}} \sum_{\mathbf{X}_s} P_s(\mathbf{X}_s) \log\left(P_s(\mathbf{X}_s)\right) \ . \end{aligned} \tag{5}$$

From a information theoretic perspective the first term in Eq.5 (with a minus sign),

$$H = -\sum_{\mathbf{X}} P(\mathbf{X}) \log\left(P(\mathbf{X})\right) \ , \tag{6}$$

quantifies the total amount of uncertainty in the system, measuring the number of bytes (if base-2 logarithms are used) necessary to define a state. The other two remaining terms in Eq.5:

$$H_m = -\sum_{c\in\mathcal{C}} \sum_{\mathbf{X}_c} P_c(\mathbf{X}_c) \log\left(P_c(\mathbf{X}_c)\right) + \sum_{s\in\mathcal{S}} \sum_{\mathbf{X}_s} P_s(\mathbf{X}_s) \log\left(P_s(\mathbf{X}_s)\right) \tag{7}$$

also quantify an uncertainty, but in this case, associated with the model of the system. In other words, by adopting the TMFG structure of interactions, H_m measures the number of bytes necessary to define the state of the system when only the interrelations among variables associated with edges in the TMFG are considered.

An algorithm to construct the TMFG with the aim of minimizing $D_{KL}(P \parallel Q)$ can be implemented by choosing at every stage the move that minimally increases H_m consistently with all other constraints. In particular, considering the TMFG construction via T_2 moves, the contribution to $D_{KL}(P \parallel Q)$ from the insertion of a vertex v added inside an existing triangular face t generating a 4-clique u is:

$$S(v,t) = \sum_{\mathbf{X}_u} P_u(\mathbf{X}_u) \log\left(P_u(\mathbf{X}_u)\right) - \sum_{\mathbf{X}_t} P_t(\mathbf{X}_t) \log\left(P_t(\mathbf{X}_t)\right) \ . \tag{8}$$

From an information theoretic perspective $-S$ is the amount of uncertainty introduced in the model by including a variable v, the TMFG structure should be constructed in a way to minimize such uncertainty. In [42] the case for normal multivariate distributions is discussed in detail.

Note that, in practice, to compute Eq.5 one must substitute the – unknown – marginal distributions of the real probability $P_c(\mathbf{X}_c)$ and $P_e(\mathbf{X}_s)$ with the corresponding empirical estimators $\hat{P}_c(\mathbf{X}_c)$ and $\hat{P}_c(\mathbf{X}_s)$ and the equality in Eq.5 would therefore become an approximate estimate. This is consistent with our approach as far as the empirical estimators are the MLE estimate of the real marginal distributions of cliques and separators.

V. EXAMPLES OF TMFG CONSTRUCTION AND COMPARISON WITH PMFG

A vast literature has demonstrated that PMFG can be used to retrieve meaningful information about the structure of interdependency in complex datasets [4, 9–12], it is therefore natural to compare the performances of the TMFG with the ones of the PMFG.

Let us first look at the scaling of execution times for TMFG and PMFG algorithms as function of the size p of the weight matrix W; results are reported in Fig.7 (seconds on a 2.6 GHz Intel Core i7®). We observe that TMFG execution times scale with the matrix dimension size p approximately as $O(p^2)$ while PMFG scales approximately as $O(p^3)$. The 2-parameters best polynomial fits give respectively: $T_{TMFG} \sim 2 \cdot 10^{-7} \cdot p^2 + 6 \cdot 10^{-4} p$ and $T_{PMFG} \sim 2 \cdot 10^{-6} \cdot p^3 + 3 \cdot 10^{-5} p^2$. Overall we can see that execution times are several orders of magnitude faster for TMFG than PMFG.

We have then compared the total retained edge weight for the following four variants of the TMFG construction:

1. TMFG: the base version of the algorithm. It uses only T_2 operators. This version of the algorithm produces chordal graphs.
2. TMFG-T1: uses T_2 followed by an optimisation stage where a number of T_1 moves are performed after every insertion of a new vertex.
3. TMFG-S: a variant of the basic algorithm with T_2 followed by local optimization with S.
4. TMFG-A: a variant of the algorithm with T_2 followed by local optimization with T_1 and A.

We have tested 9 types of random weight matrices W with different weight distributions:

1. Beta distribution with shape parameters $\alpha = 0.5$ and $\beta = 3$. This distribution is heavily skewed and is characterised by very low density on the right side of the interval $[0, 1]$.
2. Beta distribution with shape parameters $\alpha = 3$ and $\beta = 0.5$. This distribution is skewed in the opposite direction and has a high density near the right extreme of the interval $[0, 1]$.
3. Pareto distribution with power law exponent equal to 1. This distribution has a fat tail.
4. Pareto distribution with power law exponent equal to 2. This distribution has still a fat tail, but thinner than the previous one.
5. Random matrix of correlations of 400 time series generated by simulating 20 normally distributed common factors.
6. Random matrix of correlations of 400 time series generated by simulating 50 common factors. This matrix shows less structure than the one generated using 20 factors.

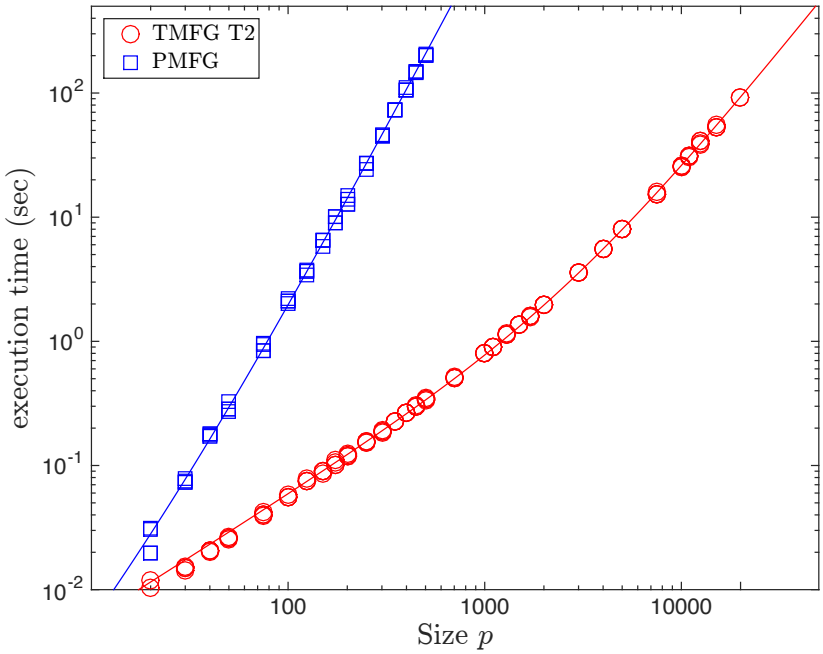

FIG. 7: Demonstration that TMFG is faster and scalable with respect to the PMFG. Comparison between execution times for TMFG and PMFG for different values of p ranging between 50 and 10000. Lines are the 2-parameters best polynomial fits (see text).

7. Random matrix of correlations of 400 time series generated by simulating 100 common factors. This matrix shows less structure than the two above.

8. Uniform distribution over $[0, 1]$.

9. Square of a real correlation matrix coefficients computed from daily log-returns of 342 US stocks, across a period of 15 years (form Jan 1997 to Jul 2012) (see [10]).

All matrices are symmetric and have size $p = 400$ except the real correlation data that have sizes $p = 342$. For all the weight matrices (excepting for the real correlation matrix) we have compared results for 100 samples. For the real correlation matrices we generated matrices by random sampling the starting point of 100 time windows of length 1000 data points over a period of 4500 points in total. Table I reports the average relative performance, defined as the ratio between the sum of the edge weight in the four variants of TMFG with respect to the sum of the edge weight in the PMFG. It shows that the PMFG is usually more effective when the density of high weights is low, while the TMFG is more effective when the density of high weights is higher or limited. This result is to be expected since the PMFG is less constrained than the TMFG in picking up isolated high-weight edges one at a time, while the TMFG is more efficient in selecting subsets of edges with a high total sum. For the random matrices of correlation we see that the TMFG performs better than the PMFG in filtering the more structured matrix generated using 20 factors. In the real case we see that the TMFG is marginally better than the PMFG. We conclude that TMFG is in general performing comparably well, and sometimes better than the

Weight matrix coefficients distribution	TMFG/ PMFG	TMFG-T1/ PMFG	TMFG-S/ PMFG	TMFG-A/ PMFG	TMFG (Time)/ PMFG (Time)
Beta(0.5,3)	95.42%	96.24%	95.72%	99.89%	0.16%
Beta(3, 0.5)	104.70%	104.73%	104.77%	104.80%	0.14%
Pareto(1)	99.97%	99.97%	99.97%	99.97%	0.17%
Pareto(2)	97.94%	98.00%	98.02%	98.32%	0.17%
Random Matrix (20 factors)	102.23%	102.63%	102.57%	103.77%	0.22%
Random Matrix (50 factors)	100.30%	100.82%	100.64%	102.54%	0.21%
Random Matrix (100 factors)	98.46%	99.14%	98.86%	101.42 %	0.21%
Uniform	116.27%	116.29%	116.34%	116.89%	0.15%
Real correlation matrix	100.11%	100.17%	100.24%	100.42%	0.15%

TABLE I: Average relative performances (ratio between sum of edge weights) of the TMFG algorithm with respect to the PMFG. Four TMFG variants and nine different weight distributions. Note that TMFG and TMFG-S are chordal graphs.

PMFG. We observe that the TMFG tends to improve relative performance as the size of the matrix increases (see Table II). When other moves are used TMFG improves performances with best performances obtained by the TMFG-A variant.

Weight matrix size p	TMFG /PMFG	TMFG-T1 /PMFG	TMFG-S /PMFG	TMFG-A /PMFG
50	88.68%	88.82%	89.35%	95.93%
100	90.49%	93.13%	92.31%	98.14%
150	92.14%	93.77%	90.44%	95.26%
300	93.73%	95.6%	94.63%	100.06%
500	96.36%	96.79%	96.6%	100.98%
700	98.83%	100.49%	98.92%	103.58%
850	98.93%	99.95%	99.99%	103.83%
1000	100.33%	100.56%	100.71%	105.39%
1200	101.34%	102.16%	101.16%	105.24%

TABLE II: Example of relative increase in performance of the TMFG algorithm with respect to PMFG when dimensionality p increases. The underlying distribution is a Beta(0.5, 3).

VI. CONCLUSIONS

We have described a new family of algorithms, TMFG, to retrieve approximate solutions to the Maximal Planar Graph problem, which have the following desirable characteristics:

1. TMFG is faster than the previously proposed method PMFG [4] with execution times that increase as the square of the weight matrix size p, while PMFG increases with the third power of p.

2. TMFG constructed with T_2 and S only produces chordal graphs. This opens the door to the use of graphical modeling. The fact that the maximum dimension of cliques is controlled by the topology entails that efficient inference algorithms can be used.

3. From TMFG construction the structure of cliques and separators is automatically retrieved.
4. The local nature of the TMFG construction allows one to model dependence using genuinely multivariate score functions over the clique elements such as Mutual Information, Total Correlation, Partial Correlation, Likelihood functions etc. (as opposed to bivariate functions such as correlation).
5. TMFG does not require preliminary calculation and sorting of all the values of the score function. Note that, in cases when the distribution is based on 3 or more variables this constraint is severe.
6. In cases where $p \gg q$ the memory footprint of the algorithm can be reduced by keeping the $p \times q$ data points in memory and calculating the dependence function on-demand, instead of the p^2 values of a correlation matrix or the p^d values of a $d-$variate dependence function.
7. The use of local and non-local operators (T_1, T_2, A, S) allows one to change the topology of the network in an easy and controlled way as the network evolves.
8. The geometric formulation of the algorithm allows one to consider generalisation to higher genus simplicial complexes structures.
9. The limited treewidth of the filtered network makes it amenable to efficient (and often exact) inference [19].

Future developments of the TMFG method are in the direction of building networks with a richer structure beyond and also below planarity while keeping a controlled complexity. We intend to apply this filtered networks to sparse modeling with application to physical, financial and biological systems. We intend to develop applications of this filtered network construction and efficient inference and use this tool to identify large scale features of financial networks in different regimes and to apply the inference algorithms to a number of problems in financial risk management, such as risk aggregation and allocation, simulation, stress testing and incorporation of non-homogeneous sources of information (such as asset prices, macroeconomic variables, and also expert opinion) into a risk management framework.

Acknowledgements

The authors acknowledge very useful discussions with Simone Severini. TA acknowledges support of the UK Economic and Social Research Council (ESRC) in funding the Systemic Risk Centre [ES/K002309/1]. TDM wishes to thank the COST Action TD1210 for partially supporting this work. GPM thanks Alun Wyn-jones for reviewing previous drafts. The authors thank Wolfram Barfuß for many useful discussions and suggestions and an anonymous referee for suggesting improvements.

[1] R. C. Prim. Shortest connection networks and some generalizations. *BellSystem Technical Journal*, 36:1389–1401, 1957.
[2] R. N. Mantegna. Hierarchical structure in financial markets. *Eur. Phys. J. B*, 11:193–197, 1999.
[3] T. Aste, T. Di Matteo, and S.T. Hyde. Complex networks on hyperbolic surfaces. *Physica A: Statistical Mechanics and its Applications*, 346(1):20–26, 2005.
[4] M. Tumminello, T. Aste, T. Di Matteo, and R. N. Mantegna. A tool for filtering information in complex systems. *Proceedings of the National Academy of Sciences of the United States of America*, 102(30):10421–10426, 2005.

[5] T. Aste. An algorithm to compute maximally filtered planar graphs. http://www.mathworks.com/matlabcentral/fileexchange/27360. Last accessed 25/08/2015.
[6] John Winston Giffin. *Graph theoretic techniques for facilities layout.* PhD thesis, University of Canterbury, 1984.
[7] Ibrahim H. Osman, Baydaa Al-Ayoubi, and Musbah Barake. A greedy random adaptive search procedure for the weighted maximal planar graph problem. *Computers and Industrial Engineering*, 45(4):635–651, 2003.
[8] Gruia Călinescu, Cristina G. Fernandes, Howard Karloff, and Alexander Zelikovsky. A new approximation algorithm for finding heavy planar subgraphs. *Algorithmica (New York)*, 36(2):179–205, 2003.
[9] W.-M. Song, T. Di Matteo, and T. Aste. Hierarchical information clustering by means of topologically embedded graphs. *PLoS ONE*, 7:e31929, 2012.
[10] Nicolo Musmeci, Tomaso Aste, and T. Di Matteo. Relation between financial market structure and the real economy: Comparison between clustering methods. PLoS ONE 10:e0126998, 2015
[11] Nicoló Musmeci, Tomaso Aste, and T. Di Matteo. Risk diversification: a study of persistence with a filtered correlation-network approach. Journal of Network Theory in Finance 1 11722, 2015.
[12] F. Pozzi, T. Di Matteo, and T. Aste. Spread of risk across financial markets: better to invest in the peripheries. *Scientific Reports*, 3:1665, 2013.
[13] Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.
[14] Amir Globerson Tommi Jaakkola. Approximate inference using planar graph decomposition. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, volume 19, page 473. MIT Press, 2007.
[15] Francis R Bach and Michael I Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems*, pages 569–576, 2001.
[16] Tomaso Aste, Ruggero Gramatica, and T. Di Matteo. Exploring complex networks via topological embedding on surfaces. *Phys. Rev. E*, 86(3):036109, 2012.
[17] J. W. Alexander. The combinatorial theory of complexes. *Ann. of Math.*, 31:294–322, 1930.
[18] Steffen L Lauritzen. *Graphical models*. Oxford University Press, 1996.
[19] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.
[20] Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric algorithms and combinatorial optimization. 1988.
[21] Mathias Drton, Bernd Sturmfels, and Seth Sullivant. *Lectures on Algebraic Statistics*, volume 39 of *Oberwolfach Seminars*. Springer, 2009.
[22] Steffen L Lauritzen, TP Speed, and K Vijayan. Decomposable graphs and hypergraphs. *Journal of the Australian Mathematical Society (Series A)*, 36(01):12–29, 1984.
[23] T. Aste and D. Sherrington. Glass transition in self organizing cellular patterns. *J. Phys. A: Math. Gen.*, 32:7049–56, 1999.
[24] Tomaso Aste, Ruggero Gramatica, and T. Di Matteo. Random and frozen states in complex triangulations. *Philosophical Magazine*, 92:246–254, 2012.
[25] Won-Min Song, T. Di Matteo, and T Aste. Building complex networks with platonic solids. *Phys. Rev. E*, 85(4):046115, 2012.
[26] Paweł Fiedor. Networks in financial markets based on the mutual information rate. *Physical Review E*, 89(5):052801, 2014.
[27] LR Foulds and DF Robinson. A strategy for solving the plant layout problem. *Operational Research Quarterly*, pages 845–855, 1976.
[28] LR Foulds and David F Robinson. Graph theoretic heuristics for the plant layout problem. *The International Journal of Production Research*, 16(1):27–37, 1978.
[29] Annegret Liebers. Planarizing graphs—a survey and annotated bibliography. *Journal of Graph Algorithms and Applications*, 5(1):74 p.–74 p., 2001.
[30] Thomas Lengauer. *Combinatorial algorithms for integrated circuit layout*. John Wiley & Sons, Inc., New York, NY, USA, 1990.
[31] Won-Min Song, Tomaso Aste, and T. Di Matteo. Correlation-based biological networks. In *Microelectronics, MEMS, and Nanotechnology*, pages 680212–680212. International Society for Optics and Photonics, 2007.

[32] David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.
[33] Takao Nishizeki and Norishige Chiba. *Planar graphs: Theory and algorithms*. Elsevier, 1988.
[34] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974.
[35] John M Boyer and Wendy Myrvold. Simplified o(n) planarity algorithms. 2001.
[36] G. L. Nemhauser, L. a. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14:265–294, 1978.
[37] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3:19, 2012.
[38] B. Dubertret, T. Aste, H. M. Ohlenbusch, and N. Rivier. Two-dimensional froths and the dynamics of biological tissues. *Phys. Rev. E*, 58(5):6368–6378, Nov 1998.
[39] J. S. Andrade Jr., H. J. Herrmann, R. F. S. Andrade, and L. R. da Silva. Apollonian networks: Simultaneously scale-free, small world, euclidean, space-filling and with matching graphs. *Phys. Rev. Lett.*, 94:018702, 2005. e-print: cond-mat/0406295.
[40] RH Green and LAR Al-Hakim. A heuristic for facilities layout planning. *Omega*, 13(5):469–474, 1985.
[41] Janny Leung. A new graph-theoretic heuristic for facility layout. *Management Science*, 38(4):594–605, 1992.
[42] Wolfram Barfuss, Guido Previde Massara, T. Di Matteo, and Tomaso Aste. Parsimonious modeling with Information Filtering Networks *http://arxiv.org/abs/1602.07349*, 2015. Last accessed 25/08/2015.
[43] W.-M. Song, T. Di Matteo, and T. Aste. Nested hierarchies in planar graphs. *Discrete Applied Mathematics*, 159:2135–2146, 2011.
[44] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.